

VLSI Systems Design Coursework

Bitslice Datapath Design

- Working individually.
- Implement an 8 bit divider *datapath* using bitslice techniques.
- Implement a synthesizable SystemVerilog *control unit* for your datapath.

Three options:

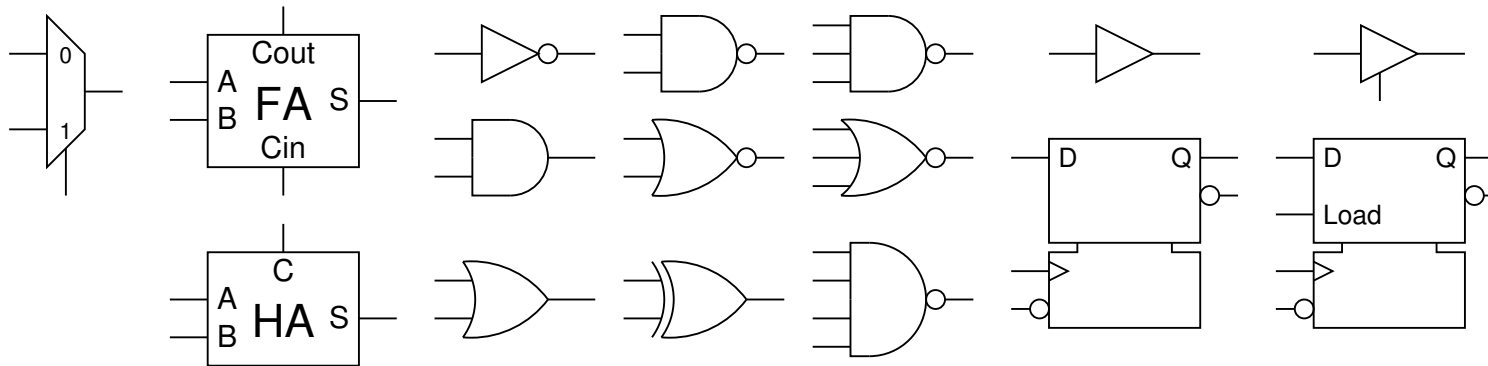
1. Datapath Architecture and Algorithm Provided, Control Signals Specified
2. Datapath Architecture and Algorithm Provided
3. Nothing Provided

For extra credit:

- Create a *complete IC* around your datapath and control unit.

Bitslice Datapath Design

Cell library¹



- Combinational Cells

`mux2 fulladder halfadder inv`
`and2 or2 nand2 nor2 xor2`
`nand3 nor3 nand4`
`buffer trisbuf`

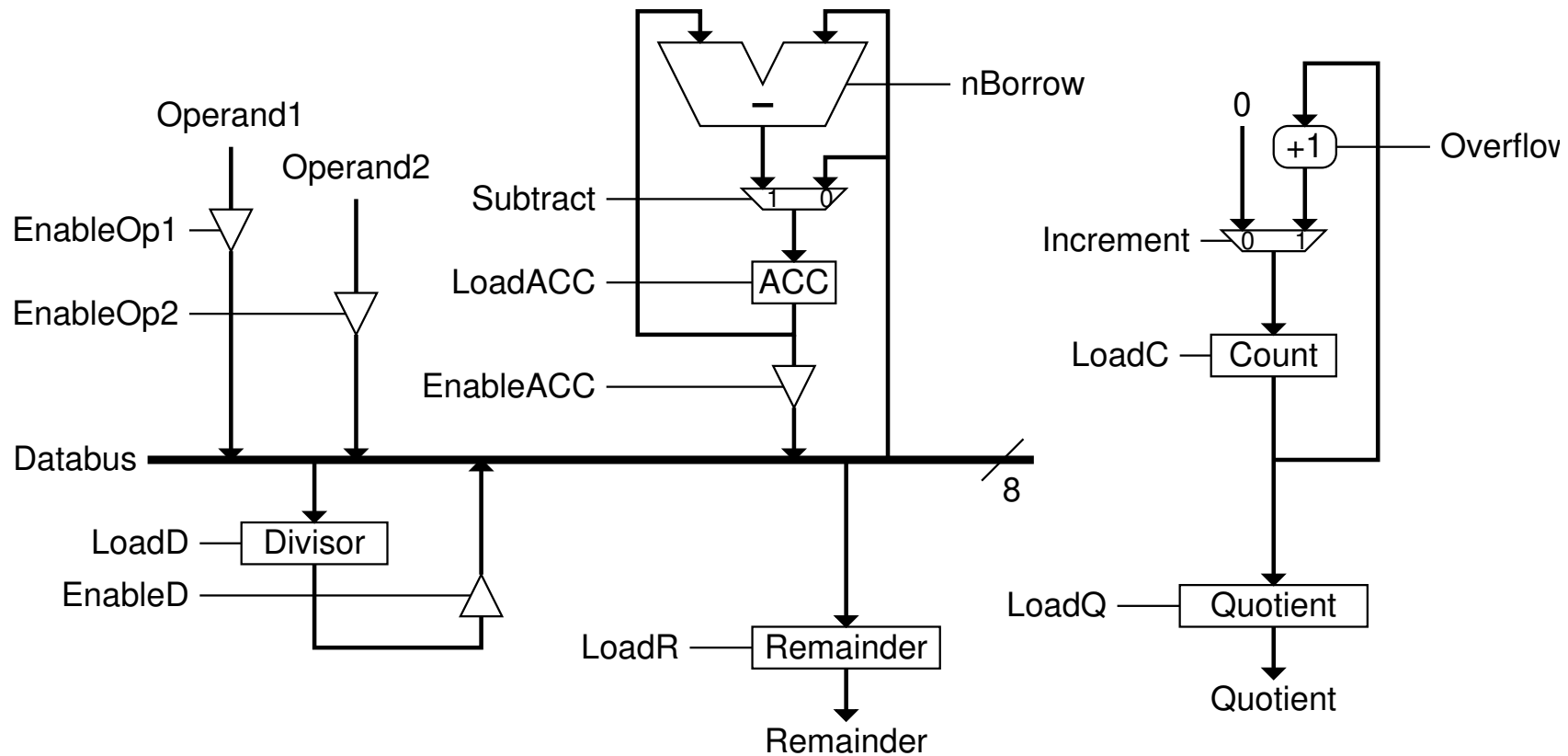
- Sequential Cells

`scandtype scanreg`

¹Please use the symbols shown when representing these cells in gate level schematics. Test and SDI connections have been intentionally omitted.

Bitslice Datapath Design

8 bit Divider Datapath - TYPE#1



Datapath Architecture

Bitslice Datapath Design

Division Algorithm - TYPE#1

- The datapath has been designed to implement the following simple division algorithm:

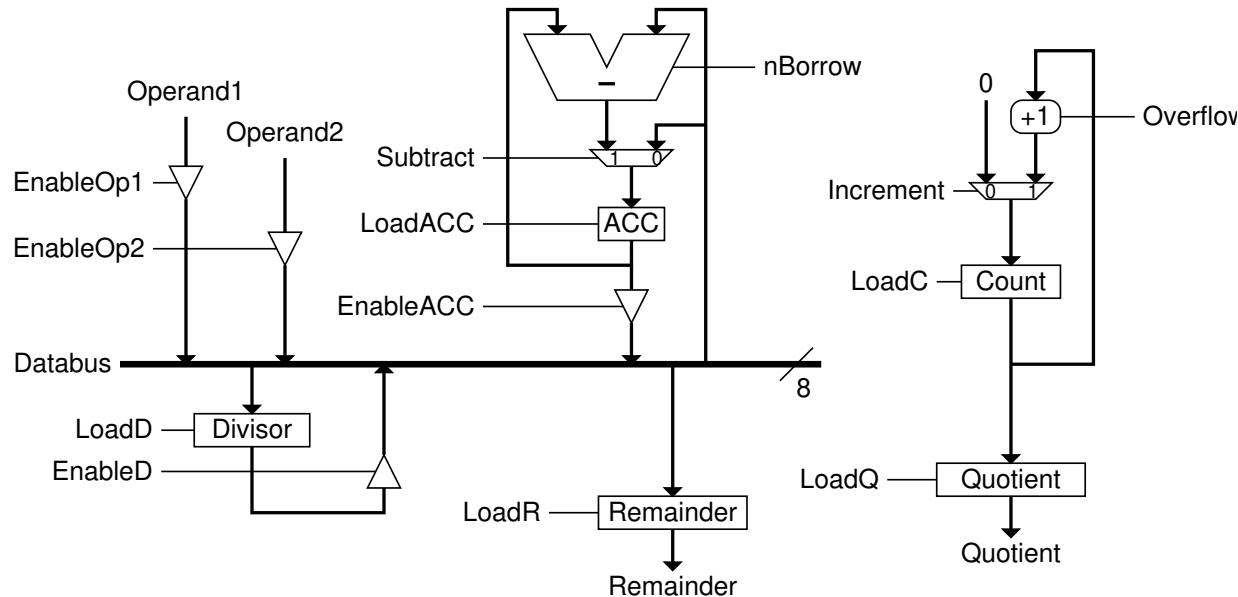
```
Count = 0;
ACC = Operand1;
Divisor = Operand2;

while ( ACC >= Divisor )
  begin
    Count = Count + 1;
    ACC = ACC - Divisor;
  end

Remainder = ACC;
Quotient = Count;
```

Bitslice Datapath Design

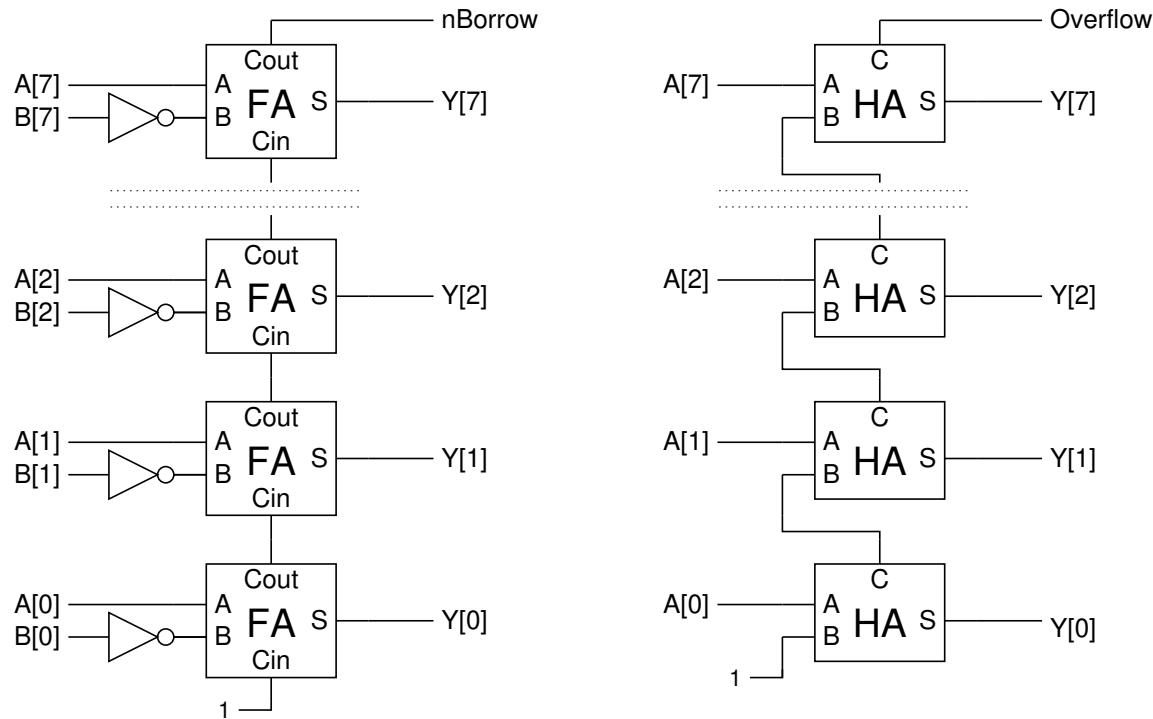
Implementaion of Blocks



- Each 8-bit register is implemented as 8 scannable register cells. The signals Clock, nReset, Test etc are not shown to keep the datapath diagram simple.
- The 8-bit subtractor and incrementer are implemented as ripple carry circuits based around full adders and half adders.

Bitslice Datapath Design

- The subtractor and incrementer² circuits:



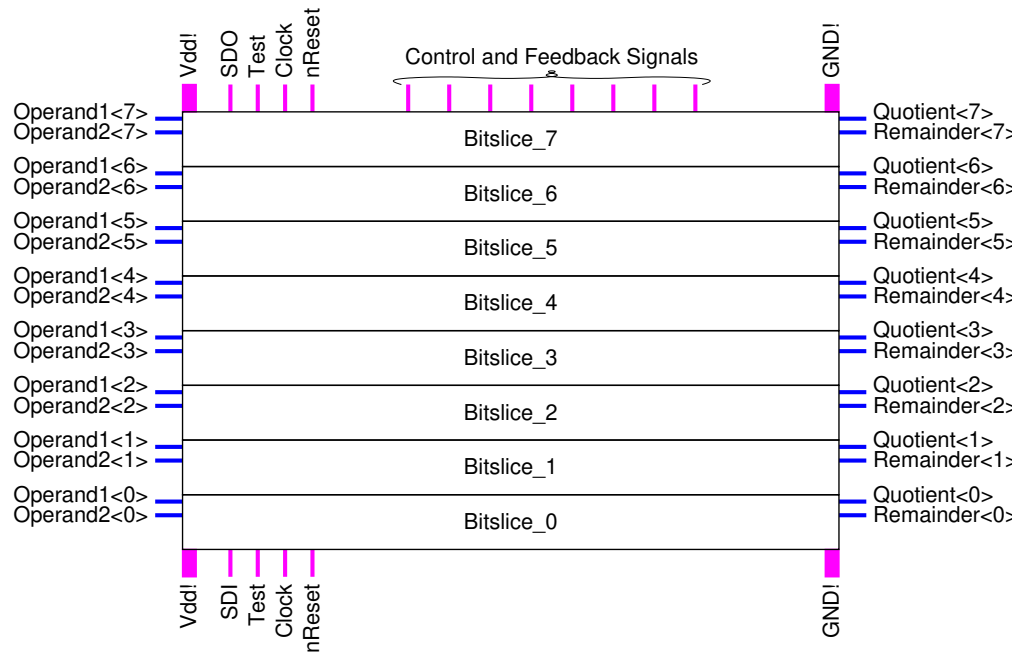
Note that the `Overflow` signal from the incrementer is not needed by the algorithm but can be used to detect *divide by zero*.

²a half adder for the incrementer can be constructed from simpler gates (or from a fulladder and a tie low) if no halfadder cell is available.

Bitslice Datapath Design

8 Identical Bitslice Cells

- all inter-bitslice wiring is by butting.



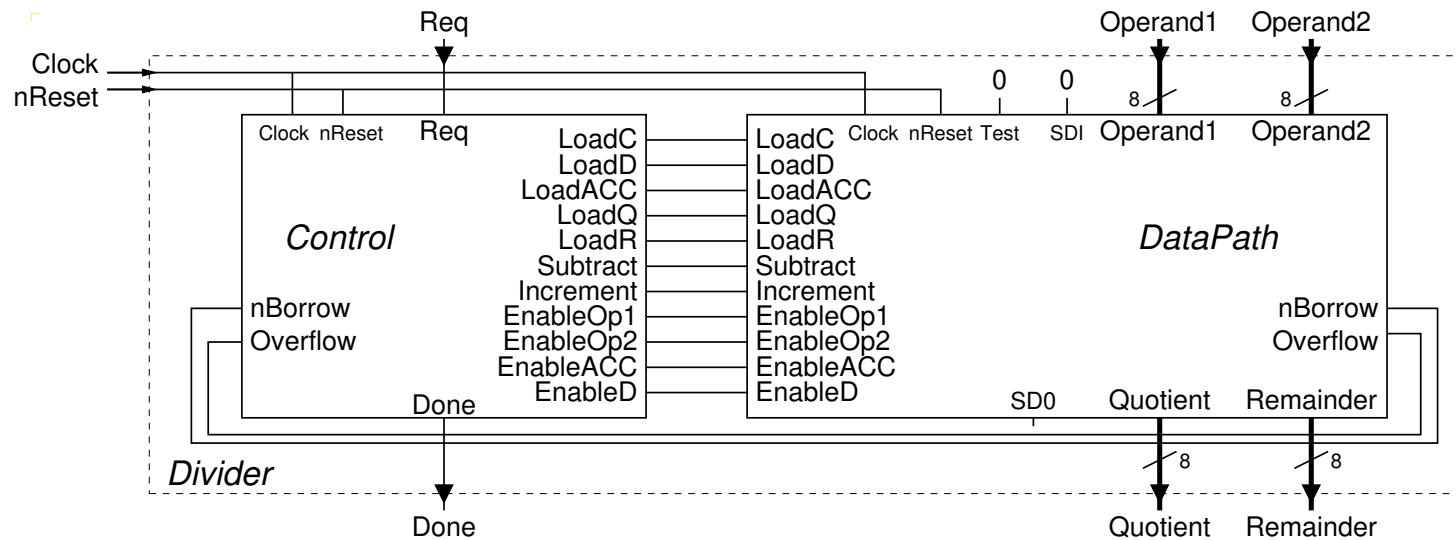
Datapath Cell

inputs: Clock, nReset, Test, SDI, Operand1, Operand2, Subtract, Increment, LoadC, LoadD, LoadACC, LoadQ, LoadR, EnableOp1, EnableOp2, EnableACC, EnableD
outputs: SDO, Quotient, Remainder, nBorrow, Overflow

Bitslice Datapath Design – System Verilog

- **control** – synthesizable behavioural model generates control signals
- **divider** – structural model instances **datapath** and **control**
- **divider_stim** – test harness.

To indicate that a new division is required, the stimulus will raise the `Req` signal for one clock cycle. Once the `Quotient` and `Remainder` become available, the **control** module will raise the `Done` signal for one cycle³.

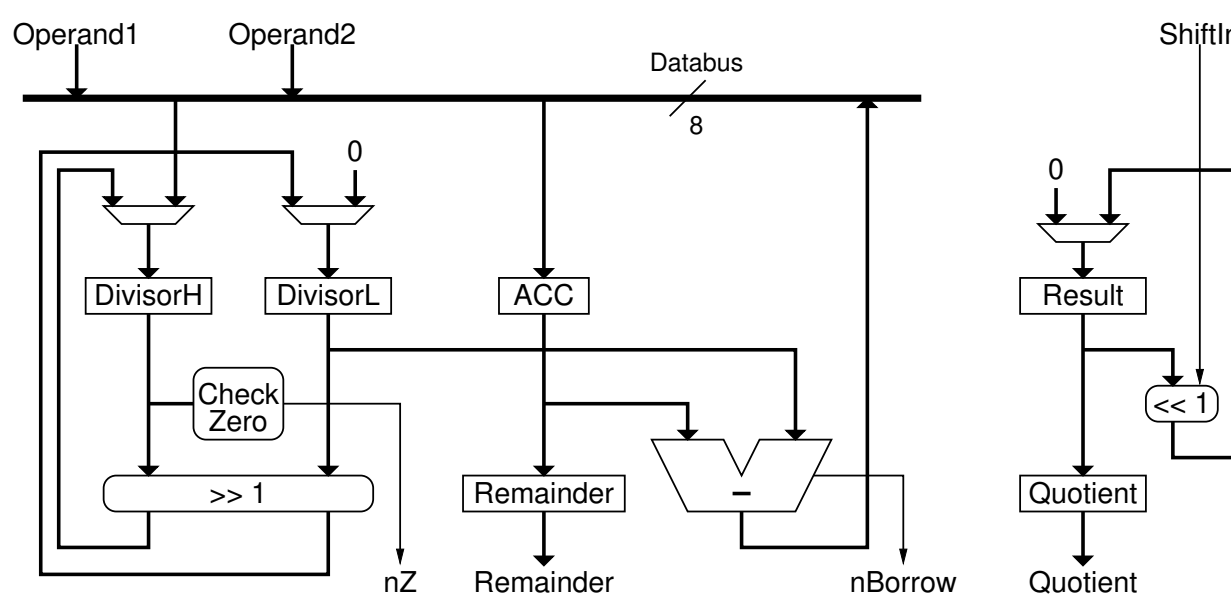


³The `Quotient` and `Remainder` values must remain stable and valid between `Done` cycles

Bitslice Datapath Design

8 bit Divider Datapath - TYPE#2

The following simplified datapath⁴ allows the implementation of a more efficient algorithm:



You can try this alternative (TYPE#2) datapath if you want to show off your talents.

⁴control signals are not specified here to allow greater flexibility in the design (i/o for complete divider will be unchanged – Req, Done, Operand1, Operand2, Quotient, Remainder, Clock, nReset)

Bitslice Datapath Design

Division Algorithm - TYPE#2

```
ACC = Operand1; Result = 0;
DivisorH = Operand2; DivisorL = 0;

for ( i = 0; i < 8; i = i+1 )
  begin
    {DivisorH,DivisorL} = {DivisorH,DivisorL} >> 1;
    Result = Result << 1;
    if ( (DivisorH == 0) && ( ACC >= DivisorL) )
      begin
        ACC = ACC - DivisorL;
        Result[0] = 1;
      end
  end
end

Remainder = ACC;
Quotient = Result;
```

Warning - think before making your life harder. With greater flexibility comes a greater chance to make mistakes. – A good working implementation of a TYPE#1 system will get more marks than an almost working implementation of a TYPE#2 system.

Bitslice Datapath Design

8 bit Divider Datapath - TYPE#3

Division Algorithm - TYPE#3

Other algorithms and datapaths may be acceptable but only if they are passed by me for the use of individual students⁵.

⁵*Type #3 designs will need to be presented for approval at the beginning of the upcoming laboratory session.*

Bitslice Datapath Design – Synthesis, Place & Route

- **control** – Synthesis

You should synthesize the control unit and verify that the synthesized gate-level version works in place of the behavioural version.

- **control** – Place & Route

You should place and route the control unit and verify that the netlist extracted from magic version works in place of the behavioural version.

- **divider** – Place & Route

As an optional extra, you can create a complete chip including the magic version of the datapath and the synthesized, placed and routed control unit.

this option will be open to students who complete the other parts of the exercise before 14:40 on the day of the final laboratory session.

Bitslice Datapath Design

Location of magic files

- Cell Library Directory

All magic leaf cells (`leftbuf.mag`, `rightend.mag`, `fulladder.mag` ...) should exist **only** in the cell library directory:

```
~/design/magic/tsmc180/cell_lib
```

- Bitslice Directory

The magic hierarchical cells (`bitslice.mag`, `datapath.mag`, `control.mag`, `divider.mag`) should exist **only** in the bitslice directory:

```
~/design/magic/tsmc180/bitslice
```

To enable access to the leaf cells when creating a hierarchical cell you will need to use the magic `addpath` command:

```
:addpath /home/<username>/design/magic/tsmc180/cell_lib
```

before using the `:getcell` command.

The easiest way to do this is to include the appropriate `addpath` command in a `".magicrc"` file in the bitslice directory. This is done automatically by the script:

```
init_bitslice_directories
```

Bitslice Datapath Design

Location of SystemVerilog files

- Behavioural Directory

- control.sv control_stim.sv

- divider.sv divider_stim.sv divider.tcl

- ~/design/magic/tsmc180/bitslice/behavioural

- Gate Level Directory

- control.sv

- ~/design/magic/tsmc180/bitslice/gate_level

- Extracted Directory

- control.vnet control.sv

- bitslice.vnet bitslice.sv bitslice_stim.sv

- datapath.vnet datapath.sv datapath_stim.sv

- divider.vnet divider.sv

- ~/design/magic/tsmc180/bitslice/extracted

Bitslice Datapath Design

Control Unit Simulation with Different Models⁶

- Behavioural

```
xmverilog behavioural/control_stim.sv \  
    behavioural/control.sv \  
    +access+r +gui
```

- Post-synthesis

```
xmverilog behavioural/control_stim.sv \  
    gate_level/control.sv -v fcdeCells.sv \  
    +access+r +gui +xmtimescale+1ns/10ps
```

- Post-layout

```
xmverilog behavioural/control_stim.sv \  
    extracted/control.sv +incdir+extracted \  
    +define+SCANPORTS +access+r +gui
```

⁶The SCANPORTS macro tells behavioural/control_stim.sv that the control module has scan ports.

Bitslice Datapath Design

Divider Simulation with Different Models⁷

```
xmverilog behavioural/divider_stim.sv behavioural/divider.sv \  
  behavioural/control.sv \  
  extracted/datapath.sv +incdir+extracted \  
  +access+r +gui +tcl+behavioural/divider.tcl
```

```
xmverilog behavioural/divider_stim.sv behavioural/divider.sv \  
  gate_level/control.sv -v fcdeCells.sv \  
  extracted/datapath.sv +incdir+extracted \  
  +access+r +gui +xmtimescale+1ns/10ps
```

```
xmverilog behavioural/divider_stim.sv behavioural/divider.sv \  
  extracted/control.sv extracted/datapath.sv +incdir+extracted \  
  +define+SCANPORTS +access+r +gui
```

```
xmverilog behavioural/divider_stim.sv \  
  extracted/divider.sv +incdir+extracted \  
  +define+SCANPORTS +access+r +gui
```

⁷The SCANPORTS macro tells behavioural/divider_stim.sv that the divider module has scan ports and it tells behavioural/divider.sv that the control module has scan ports.

Bitslice Datapath Design

Submission – Files

Prepare script (`prepare_vlsi_desex4`) will collect the magic and SystemVerilog files into a single `handin.tar` file.

The script will also run a number of simple simulations.

Deadline for file submission is 16:00 on Friday 10th January.

Submission – Supporting Figures

System design diagrams (in each case **all** control signals must be shown):

- Datapath Architecture Diagram *single A4 page PDF document readable at A5 size*
- Bitslice Gate Level Schematic *single A4 page PDF document readable at A5 size*
- ASM Chart *single A4 page PDF document readable at A5 size*
- All 3 figures combined on one page *single A4 page PDF document*

Deadline for figure submissions is 16:00 on Friday 17th January.

