Write a test fixture:

```c
output = 0;
input [3:0] int1, int2;
input clock;

module sign_compare (clock, int1, int2, result);

// result = 1 if int1 > int2; int1, int2 are signed 2's complement
comparison output in a register

You have written a module that compares 2 4-bit signed numbers and stores the
2's complement compare logic:

Verification Example
```
primitive primdff(q,cp,d);
/* ************************************************************************* *
 /* Copyright (c) 1994 Advanced Microelectronics Division (ITD/AuE)*/
 /* */
 /* This material may be reproduced by or as permitted by the U.S. */
 /* Government pursuant to the copyright license under the clause */
 /* at DFARS 252.227-7013 (November 1994) */
 /* */
 /* This file contains data and information of a proprietary */
 /* nature to ITD/AuE that is unpublished and should not be */
 /* disclosed to third parties outside the US Government, or */
 /* as permitted by the US Government, without the written */
 /* consent of the ITD Advanced Microelectronics Division or */
 /* it's representative. */
 /* */
 /* Institute for Technology Development */
 /* Advanced Microelectronics Division */
 /* PO Box 55729 */
 /* Jackson, MS 39296-5729 */
 /* (601) 932-7620 */
 /* */
 /* These notices shall not be removed and shall be included on */
 /* any reproduction here-of */
 /* */
 /* ************************************************************************* */

output q;
reg q;
input cp,d;
table

/* user defined primitive "primdff" as a table -- useful for */
/* accelerated simulation and compiling a synthesis library */

//
// cp  d  q
// clocking data on the rising edge
r 1 : q : 1;
r 0 : q : 0;
// ignoring the falling edge of the clock
n ? : q : -;
// ignoring the edges on data
* 0 : q : 0;
* 1 : q : 1;
? * : q : -;
endtable
endprimitive

Unspecified transistors
⇒ Don't care CP/P
⇒ last 3 lines
(NB: like 'case'
− compared in sequence)
Identify Control points

PE

S1 S2 mux

LA-B1

CL

new Dist

Accumulate In

2) Design Data path
   -> implement