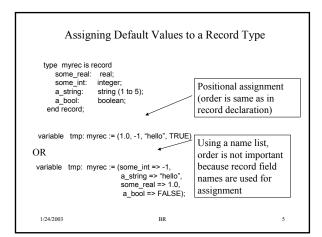
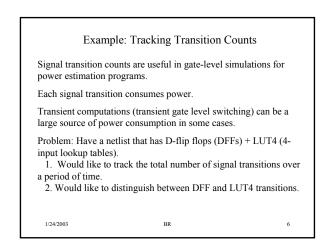
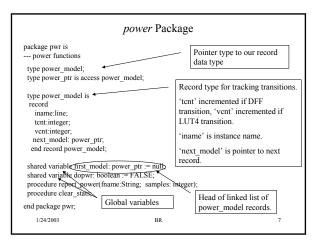
utilities directory h types/functions/pro modeling TYPE Bit_Memory IS ARR/ TYPE Nibble_Memory IS ARR TYPE Byte_Memory IS ARR TYPE Word_Memory IS ARR	Yype Definition Examples has a package called <i>memory</i> to ocedures that are useful for me AY (Natural RANGE ↔) OF Std_Ulogic; RAY (Natural RANGE ↔) OF Std_Ulogic, RAY (Natural RANGE ↔) OF Std_Ulogic, NRRAY (Natural RANGE ↔) OF Std_Ulogic	vector(3 DOWNTO 0); vector(7 DOWNTO 0); vector(15 DOWNTO 0);	Multi This is an example of a mult TYPE Byte_Memory IS ARRAY (Natura unconstrained Only one array index rang i.e. "Natural Range $<$ " The other ranges must be of type as: TYPE A_Memory IS ARRAY (Natura	al RANGE ⇔) OF Std_Ulogic_Vec constraine ge can be unconstrained constrained. Would be	tor(7 DOWNTO 0); ed
1/24/2003	BR	1	1/24/2003	BR	2

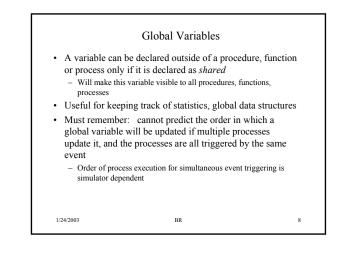
Array Assignments				
When assigning one array to another, the <i>slice</i> size must be the same as well as the data type.				
TYPE Byte_Memory IS ARRAY (Natural RANGE <>) OF Std_Ulogic_Vector(7 DOWNTO 0); TYPE Word_Memory IS ARRAY (Natural RANGE <>) OF Std_Ulogic_Vector(15 DOWNTO 0);				
variable a_mem: Byte_Memory(0 to 1023); variable b_mem: Byte_Memory(0 to 2047); variable c_mem : Word_Memory (0 to 511);				
a_mem (3 to 10) := b_mem (11 to 18);				
a_mem (20 to 30) := b_mem (20 to 40); illegal, slice size is different				
c_mem(2) := a_mem (2);				
1/24/2003 BR 3				

RECORD types	
A record type is a composite object type whose elements ar	e
named:	
type myrec is record	
some_real: real;	
some_int: integer;	
a_string: string (1 to 5);	
a_bool: boolean;	
end record;	
Usage example:	
variable tmp; myrec;	
tmp.some_real := -30.4	
tmp. some_int := 10 ;	
tmp.a_string := "Hello"; tmp.a_bool := TRUE;	
Signals can be record types!!! Can be helpful for complex	
modeling. 1/24/2003 BR	4









Modeling Approach

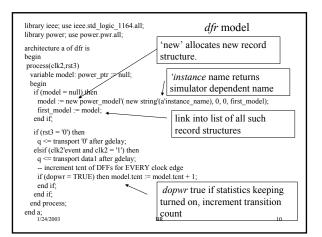
- At startup, each instance in our netlist will create a record of type 'power model'.
 - Insert this into a linked list of all such power_model records
 - A global shared variable will be used to point to the head of this
 - linked list
- Each time a signal transition occurs on an input, increment a counter in the power_model
 - For DFFs, increment 'tcnt'
 - For LUT4s, increment 'vcnt'.
- Can enable/disable transition counter via a global variable called *dopwr*

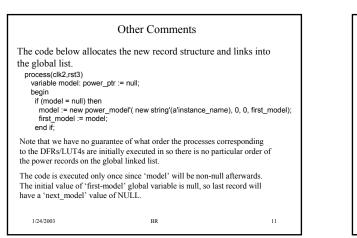
9

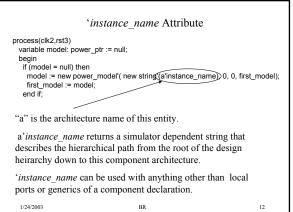
- Only increment transition counts if this variable is TRUE
- · Print transition stats using 'report_power' procedure
- Clear stats using 'clear_stats' procedure

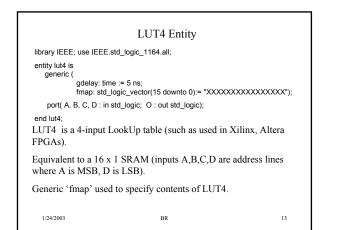
1/24/2003

BR



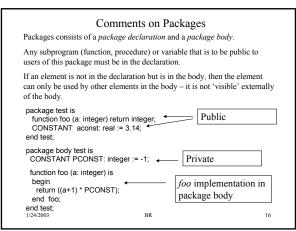


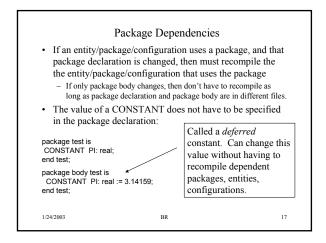


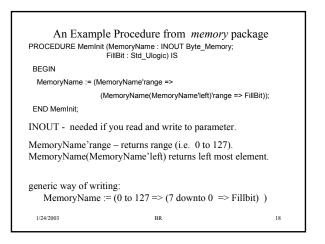


library IEEE; use IEEE.std_logic_1164.all;				
library power;use power.pwr.all;	LUT4 Architecture			
architecture a of lut4 is	Lor4 / fieliteeture			
begin				
process (A,B,C,D)				
variable index, lastval:integer; variable lasttrig: time := 0 ns;				
variable model: power ptr := null;				
begin				
if (model = null) then				
model := new power_model'(new string'	(a'instance_name),0, 0, first_model);			
first_model := model;				
	Compute LUT4 address			
index := 0;				
	end if; Only count transition if			
	end if; end if: current address is different			
	end if; from last address. Filter			
O <= transport fmap(index) after gdelay; spikes < 1 ns.				
if (lastval /= index and ((now - lastrig) > 1 ns)) then if (dopwr = TRUE and (not nopower)) then model.vcnt := model.vcnt + 1; end if;				
lastval := index; lasttrig := NOW;				
end if;				
end process; end a;	BR 14			

Traversing the Record List	
<i>clear_stats</i> procedure is used to zero out statistics after recording some signal transitions.	
procedure clear_stats is variable head_ptr: power_ptr; begin head_ptr := first_model; while head_ptr /= null loop head_ptr.tcnt :=0; head_ptr.cnt :=0; head_ptr := head_ptr.next_model; end loop; end;	
<i>report_power</i> procedure traverses list in a similar fashion except it sums the transition counts and prints out values to screen	
1/24/2003 BR	15







An Example <i>function</i> from <i>memory</i> package				
FUNCTION MemRead (MemoryName Address : Str RETURN Std Ulogic Vector	d_Ulogic_\			
BEGIN	- • -			
IF (Is_X(Address)) THEN		contains an 'X'?		
RETURN (MemoryName(MemoryName'left)'range => 'X');				
ELSE				
RETURN (MemoryName(To_Integ	er(Address	s)));		
END IF;				
END MemRead;		pe conversion to		
		eger type since index		
	15	of type NATURAL		
1/24/2003	BR	19		

An Example procedure from memory package				
PROCEDURE MemWrite (MemoryName : INOUT Byte_Memory; Address : Std_Ulogic_Vector; Data : Std_Ulogic_Vector) IS				
BEGIN				
IF (Is_X(Address)) THEN NULL; ELSE MemoryName(To_Integer(Address)) := Data; END IF; END MemWrite;	Assign data to specified memory location.			
1/24/2003 BR	20			