

Synopsys Behavioral Compiler

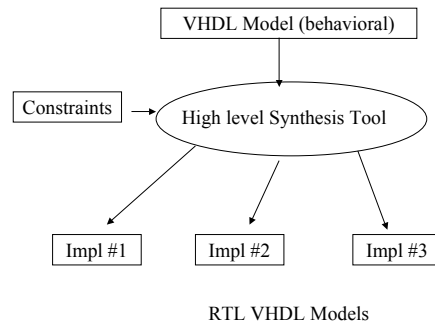
- Synopsys Behavioral Compiler
 - Synthesis tool to quickly explore area/time tradeoffs in datapath designs.
- Example: How many ways can you build a datapath that adds 4 numbers?
 - Use 1 adder, takes 4 clocks (1 clk for input, 3 for sums)
 - Use 2 adders, takes 3 clocks
 - Using pipelining and 3 adders, can get a throughput of 1 new result every clock

2/26/2002

BR

1

High Level Synthesis



2/26/2002

BR

2

Behavioral Code

- The VHDL code written for behavioral model is very restricted in both the types of statements used and the style that is used
- The examples shown here are for Synopsys Behavioral Compiler
- Coding style shown *must* be followed in order for synthesis tool to operate correctly

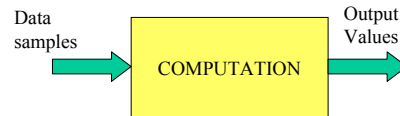
2/26/2002

BR

3

Definitions

- *Latency* – the number of clocks from an input value to the corresponding output value
- *Initiation Rate* – the rate at which new input values are accepted (measured in clock cycles).

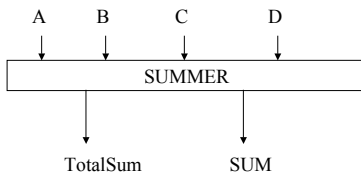


2/26/2002

BR

4

summer.vhd



Each computation will input four values (A,B,C,D) and output SUM = A+B+C+D. TOTALSUM will be a running sum of all values entered.

2/26/2002

BR

5

Synopsys Behavioral Compiler

- The zip archive attached to this lecture has the 'summer' example
 - Each sample time, add 4 numbers together
 - Also keep a running sum
- Implementations
 - l0_p0 -- minimum area, no constraints
 - l4_p4 -- 4 clocks per sample period, no pipelining
 - l4_p2 -- 4 clock latency, 2 clock initiation rate
 - l4_p1_v1 -- 4 clock latency, 1 clock initiation rate (1 super state)
 - l4_p1_v2 -- 4 clock latency, 1 clock initiation rate (2 super states)

2/26/2002

BR

6

ZIP Archive Directory Structure

```
./vhdl/src/bc -- vhdl code for simulation
./synopsys/bc -- main directory for Synopsys execution
./synopsys/bc/behv -- contains behavioral code Synopsys BC
./synopsys/bc/gate -- resulting gate level implementations placed here
./synopsys/bc/*.script -- script files for running synopsys bc (summer_10_p0.script, summer_14_p2.script, etc)
./synopsys/bc/*.rpt -- report files
```

2/26/2002

BR

7

synopsys/bc/behv/summer.vhd

```
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity summer is
    port (a,b,c,d: in std_logic_vector(7 downto 0);
          clk: in std_logic;
          reset: in std_logic;
          input_rdy : out std_logic;
          output_rdy : out std_logic;
          totalsum : out std_logic_vector(7 downto 0);
          sum: out std_logic_vector(7 downto 0)
    );
end summer;
```

Entity
Four 8-bit input ports.
Two 8-bit output ports

2/26/2002

BR

8

Architecture

architecture behv of summer is
begin

```
main:process
variable va,vb,vc,vd: signed(7 downto 0);
variable vx,vy,vz: signed(7 downto 0);
variable vtotalsum,vsum: signed(7 downto 0);
begin
reset_loop: loop
-- initialize variables vtotalsum := "00000000";
output_rdy <= '0'; input_rdy <= '0';
wait until clk'event and clk = '1';
if (reset = '1') then exit reset_loop; end if;
input_rdy <= '1';
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;
l1: loop
```

Outer loop is reset loop. All initialization code goes here.

2/26/2002 see next page

BR

9

Architecture (cont)

```
l1: loop
input_rdy <= '0'; output_rdy <= '0';
va := unsigned(a); vb := unsigned(b);
vc := unsigned(c); vd := unsigned(d);
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;
input_rdy <= '1'; output_rdy <= '1';
vx := va + vb; vy := vc + vd;
vsum := vx + vy;
vtotalsum := vtotalsum + vsum;
sum <= std_logic_vector(vsum);
totalsum <= std_logic_vector(vtotalsum);
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;
end loop; -- L1
end loop; -- reset_loop
end process;
end behv;
```

Inner loop is computation loop.

2/26/2002

BR

10

Super States

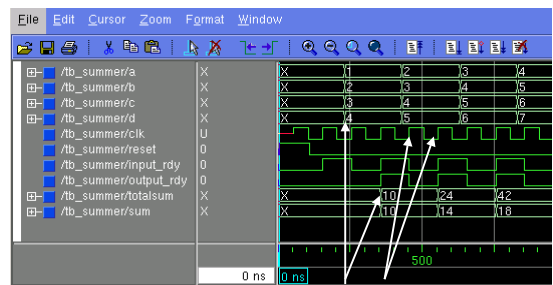
- Each 'wait' statement in the outer/inner loop is called a *super state*
- When BC is generating an implementation, it may use 1 or more clock cycles for a super state based on the constraints
- The minimum amount of latency for a loop will be equal to the number of wait states in the loop
 - Other constraints may cause this minimum latency to increase

2/26/2002

BR

11

summer.vhd simulation (before synthesis)



$1+2+3+4 = 10$

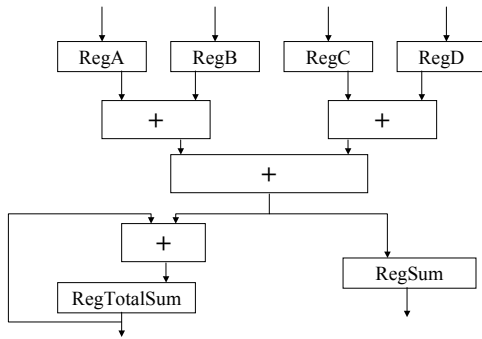
Two clocks, one for each super state

2/26/2002

BR

12

Normal logic synthesis using the *summer.vhd* code produces the following architecture:



2/26/2002

BR

13

Running Synopsys BC

To run Synopsys BC for a test case, do:

```
% swsetup synopsys
% cd synopsys/bc
% dc_shell -f summer_10_p0.script
```

Gate level implementation placed in *gate/summer_10_p0.vhd*

Report file in *summer_10_p0.rpt*

2/26/2002

BR

14

Constraints

- Constraints for BC are clock period, latency, initiation rate
- Clock period will control what types of compute elements are used (i.e., fast adder structures vs slow adders)
 - Requires a library that is characterized for timing, we will always just use a slow clock period
- Latency – how many clocks from an input value to the corresponding output value
 - the number of clocks that the compute loop will take
- Initiation Rate – number of clocks between new input values
 - If initiation rate = latency, no pipelining is being done.
 - Will discuss this in more detail later.

2/26/2002

BR

15

summer_10_p0.script

```
target_library = class.db
link_library = class.db
bc_enable_analysis_info = true
analyze -f vhdl behv/summer.vhd
elaborate -schedule summer
create_clock clk -p 1000
bc_check_design -io superstate_fixed
bc_time_design -force
dont_chain_operations -from -into main/reset_loop/l1/add*
set_common_resource main/reset_loop/l1/add*
/* constraints go here */
schedule -extend_latency -io superstate_fixed
report_schedule -summary -verbose_fsm -operations -variables >
summer_10_p0.rpt
target_library = gtech.db
compile
vhdlout_equations = true
vhdlout_dont_write_types = true
vhdlout_use_packages = {ieee.std_logic_1164, ieee.std_logic_arith}
vhdlout_architecture_name = syn_10_p0
write -hierarchy -f vhdl -output gate/summer_10_p0.vhd
sh filter_entity.pl summer gate/summer_10_p0.vhd
quit
```

Annotations in the original image:

- read file (points to `analyze`)
- long clk period (points to `create_clock`)
- no chaining, each adder separated by register (points to `dont_chain_operations`)
- minimum area, no constraint on compute latency (points to `schedule -extend_latency`)
- write result (points to `write -hierarchy`)

2/26/2002

BR

16

summer_10_p0.rpt

- Report file gives a list of hardware resources used
- Schedule for hardware resources
- Type of hardware resources

Timing Summary

Clock period 1000.00

Loop timing information:

```
main.....7 cycles (cycles 0 - 7)
reset_loop.....7 cycles (cycles 0 - 7)
l1.....5 cycles (cycles 2 - 7)
```

inner loop 5 clocks

2/26/2002

BR

17

summer_10_p0.rpt (cont)

```
-----
Resource types
-----
Register Types
-----
8-bit register.....3
Operator Types
-----
(8_8->8)-bit DW01_add.....1
I/O Ports
-----
1-bit registered output port.....2
8-bit input port.....4
8-bit registered output port.....2
```

Annotations in the original image:

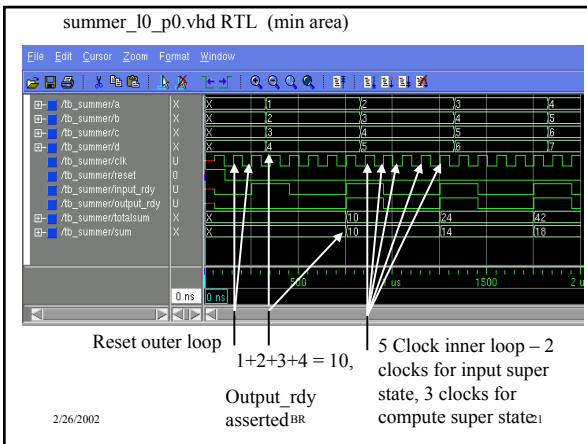
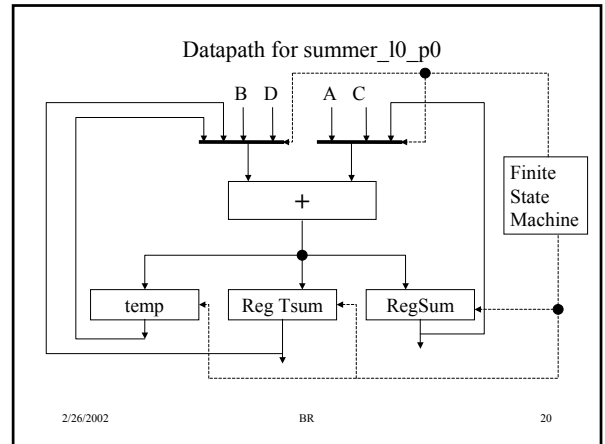
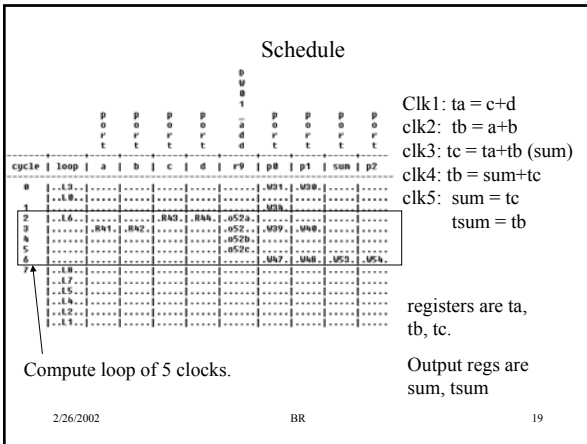
- Three 8-bit registers for internal values (points to `8-bit register.....3`)
- One 8-bit adder (points to `(8_8->8)-bit DW01_add.....1`)
- Two 8-bit registered output ports (points to `8-bit registered output port.....2`)

- The datapaths that BC generates places registers on all of the outputs
 - Input values are assumed stable throughout the superstate in which they are used. Input values are NOT registered.

2/26/2002

BR

18



summer_14_p4.script

Require compute loop to complete in 4 clocks. Only difference in script is constraints:

```
/* constraints go here */

/* Use this constraint if No pipelining, but limiting loop time */
set_cycles 4 -from_beginning main/reset_loop/1 -to_end main/reset_loop/11
schedule -io superstate_fixed
```

Set compute loop time to 4 clocks, dropped '-extend-latency' from schedule command.

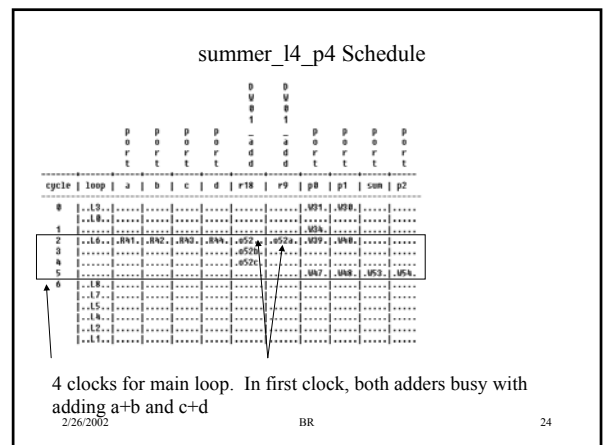
2/26/2002 BR 22

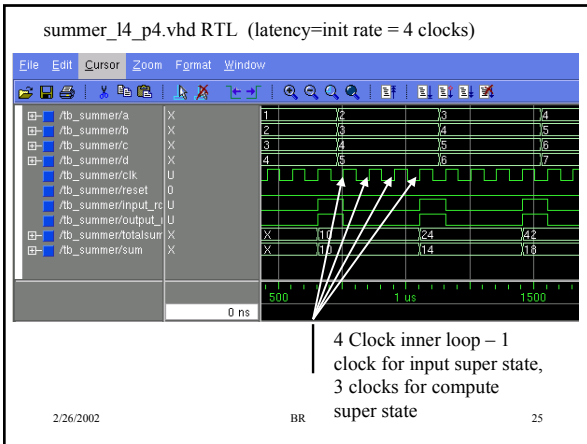
summer_14_p4 resources

Timing Summary	
Clock period 1000.00	
Loop timing information:	
main.....	6 cycles (cycles 0 - 6)
reset_loop.....	6 cycles (cycles 0 - 6)
l1.....	4 cycles (cycles 2 - 6)
Register Types	
8-bit register.....3	
Operator Types	
(8_8->8)-bit DW01_add.....2	
I/O Ports	
1-bit registered output port.....2	
8-bit input port.....4	
8-bit registered output port.....2	

4 cycles for loop, but need an extra adder.

2/26/2002 BR 23





summer_14_p2 -- Pipelining

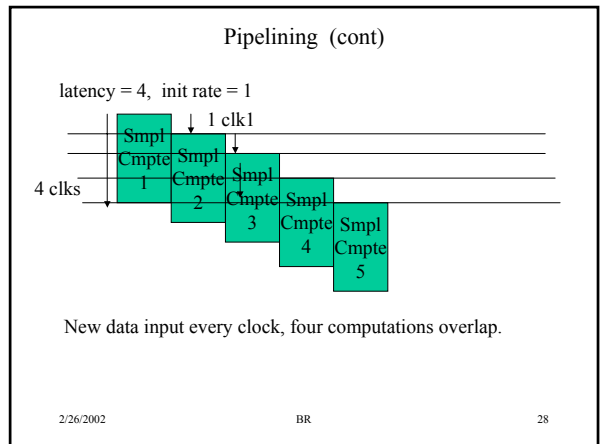
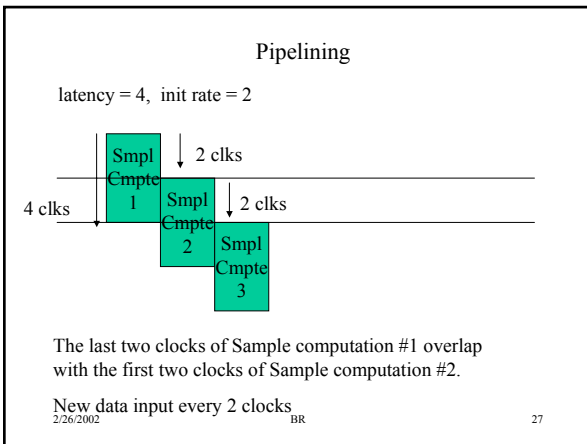
Pipelining implies overlapped computation of sample periods.

To generate pipelining, must specify an initiation rate that is less than than the latency.

Also, the initiation rate must be evenly divisible into the latency because this defines how many sample computations will be overlapped:

latency = 4, init rate = 2, $4/2 = 2$ overlapped loop bodies
 latency = 4, init rate = 1, $4/1 = 4$ overlapped loop bodies

2/26/2002 BR 26



summer_14_p2.script

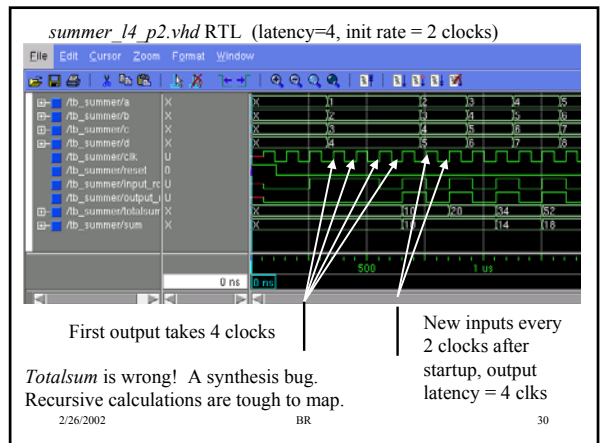
Require compute loop to complete in 4 clocks but also specify an initiation rate = 2.

```
/* constraints go here */

/* use this constraint if want to do pipelining */
pipeline_loop main/reset_loop/1 -latency 4 -initiation_interval 2
schedule -io superstate_fixed
```

Use *pipeline_loop* command.
 Set compute loop time to 4 clocks
 Initiation rate to 2.

2/26/2002 BR 29



Handshaking

- Used *input_rdy* and *output_rdy* signals in design so that one testbench could be used for all designs (l0_p0, l4_p4, l4_p2)
 - Input Rdy toggles high and low for each sample
- However, for initiation rate = 1, need *input_rdy* to stay high once compute loop is entered because we get a new value each clock
 - Had to write a new testbench (tb_summer_pipe) and new behavioral models (summer_pipe1.vhd, summer_pipe2.vhd)
 - Two different versions of fully pipelined (1 clock per input sample) case are provided

2/26/2002

BR

31

summer_pipe1.vhd

This file used for latency = 4, init rate = 1 case. Input Rdy does not toggle, only 1 super state in loop.

```

reset_loop: loop
-- initialize variables
vtotalsum := "00000000";
output_rdy <= '0'; input_rdy <= '0';
wait until clk'event and clk = '1';
input_rdy <= '1';
wait until clk'event and clk = '1';
if (reset = '1') then exit reset_loop; end if;
l1: loop
va := unsigned(a); vb := unsigned(b);
vc := unsigned(c); vd := unsigned(d);
vx := va + vb; vy := vc + vd; vsum := vx + vy;
vtotalsum := vtotalsum + vsum;
sum <= std_logic_vector(vsum);
totalsum <= std_logic_vector(vtotalsum);
output_rdy <= '1';
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;
end loop; -- l1
    
```

only one superstate, behavioral simulation matches gate level simulation

2/26/2002

BR

32

summer_pipe2.vhd

Two super states in loop. Behavioral simulation will not match gate level, just need to be aware of this.

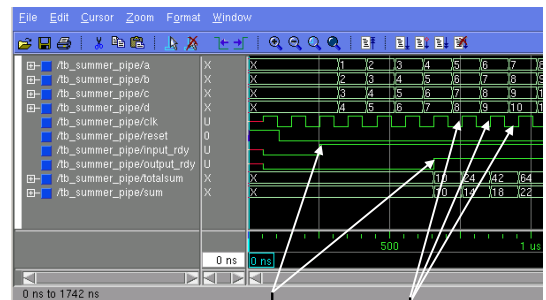
```

reset_loop: loop
-- initialize variables
vtotalsum := "00000000";
output_rdy <= '0'; input_rdy <= '0';
wait until clk'event and clk = '1';
if (reset = '1') then exit reset_loop; end if;
input_rdy <= '1';
wait until clk'event and clk = '1';
if (reset = '1') then exit reset_loop; end if;
l1: loop
va := unsigned(a); vb := unsigned(b);
vc := unsigned(c); vd := unsigned(d);
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;
vx := va + vb; vy := vc + vd;
vsum := vx + vy;
vtotalsum := vtotalsum + vsum;
sum <= std_logic_vector(vsum); totalsum <= std_logic_vector(vtotalsum);
output_rdy <= '1';
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;
end loop; -- l1
    
```

2/26/2002

33

summer_l4_p1_v2.vhd RTL (latency=4, init rate = 1 clocks)



Input, Output ready signals do not toggle.

New input/output values every clock

2/26/2002

BR

34

Resource Summary

Design	#Adders	#8-bit Registers	Latency	IRate
l0_p0	1	5	5	5
l4_p4	2	5	4	4
l4_p2	3	7	4	2
l4_p1_v1	4	9	4	1
l4_p1_v2	4	9	4	1

2/26/2002

BR

35

VHDL Files in Archive

- All VHDL files under src/bc
- Configurations for each case
 - cfg_summer_behv (behavioral, non-pipelined),
 - cfg_summer_l0_p0, cfg_summer_l4_p2, etc...
- Two different test benches
 - tb_summer.vhd for all cases except init rate = 1
 - tb_summer_pipe.vhd for initiation rate = 1

2/26/2002

BR

36