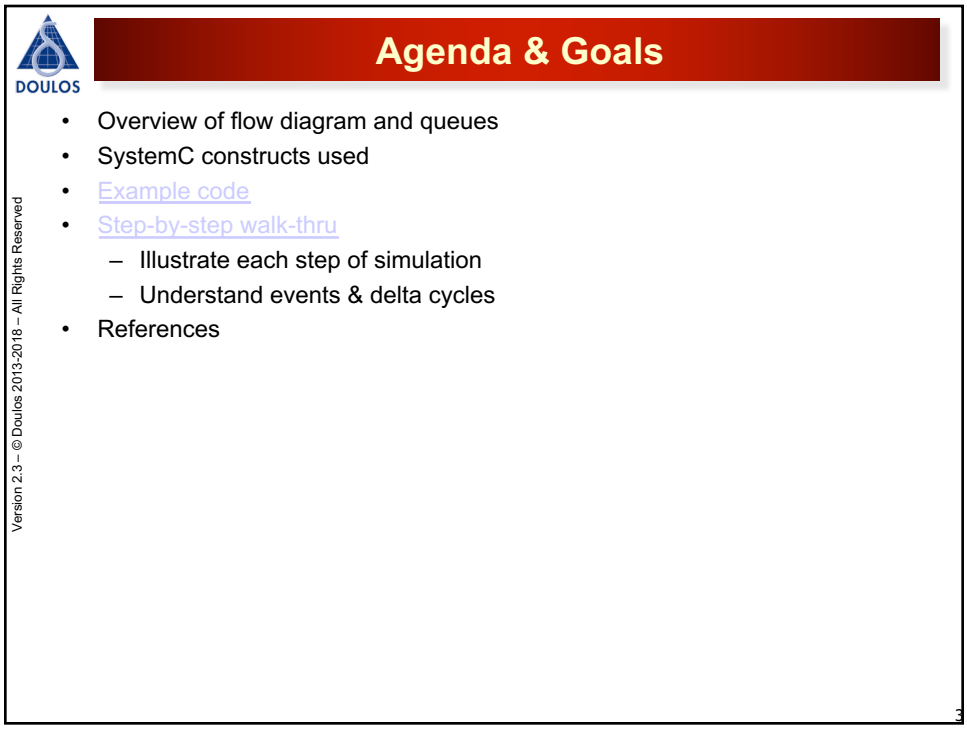
The SystemC Simulation Engine

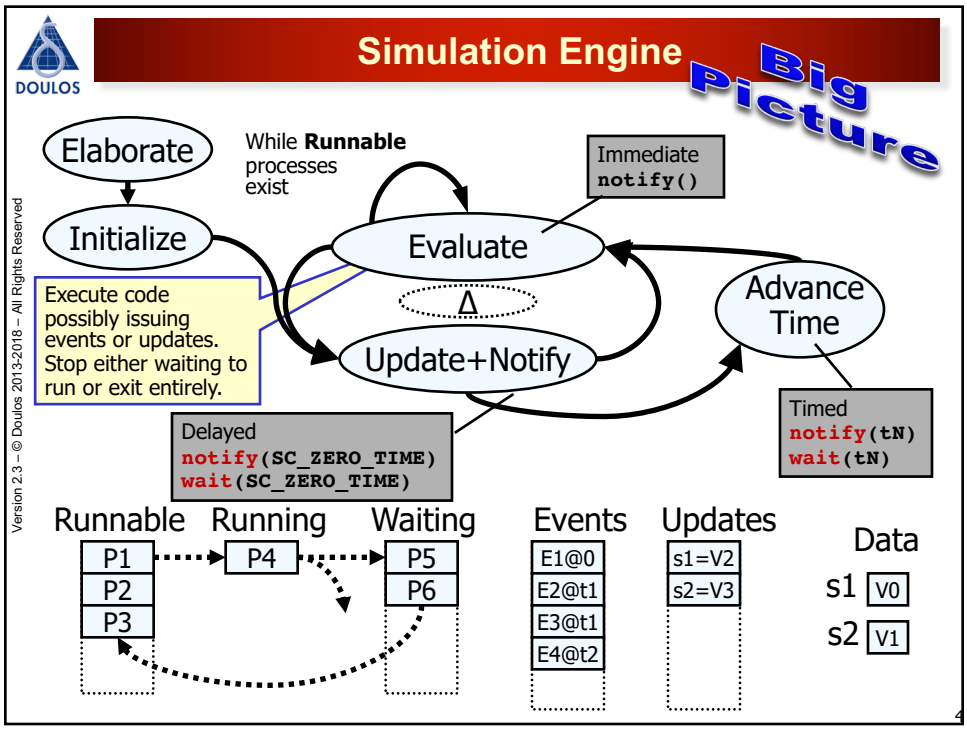
An Interactive Exploration of an Event Driven Simulator

Doulos Inc.
David C. Black, Senior Member Technical Staff

Copyright Permissions

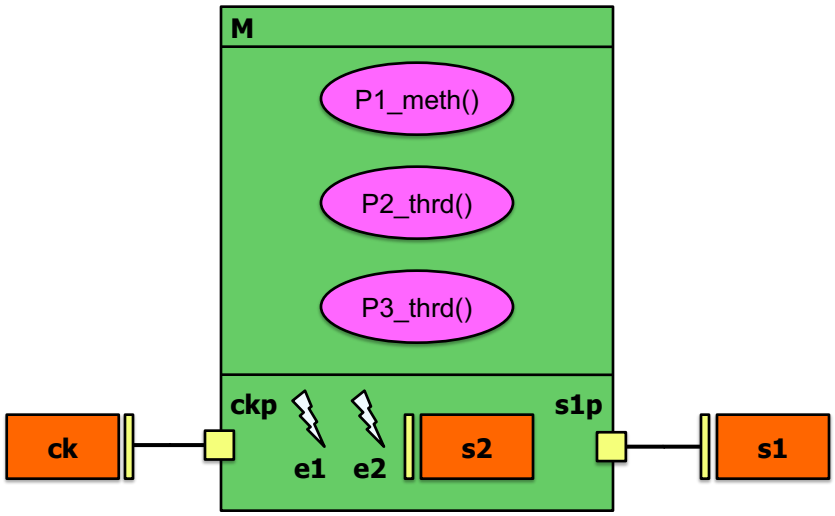
- This work is copyright 2013-2018 by Doulos Inc. with all rights reserved. Permission is hereby granted for professors of recognized licensed Colleges and Universities to use in lectures including the creation of handouts for registered students. Individuals may download and use for personal education, but not for display, public presentation nor redistribution electronically or in any other format. Any other commercial use is strictly prohibited without explicit written permission from Doulos Inc.
- In simpler language, competitors to Doulos training services may not use this material unless separately licensed. Display on the Internet is prohibited except from Doulos authorized locations.


DOULOS
Agenda & Goals
• Overview of flow diagram and queues
• SystemC constructs used
• Example code
• Step-by-step walk-thru
– Illustrate each step of simulation
– Understand events & delta cycles
• References
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
3


DOULOS
Simulation Engine
Big Picture
Elaborate
Initialize
While Runnable processes exist
Evaluate
Δ
Update+Notify
Advance Time
Immediate notify()
Execute code possibly issuing events or updates. Stop either waiting to run or exit entirely.
Delayed
notify(SC_ZERO_TIME)
wait(SC_ZERO_TIME)
Timed
notify(tN)
wait(tN)
Runnable
Running
Waiting
P1
P2
P3
P4
P5
P6
Events
E1@0
E2@t1
E3@t1
E4@t2
Updates
s1=V2
s2=V3
Data
s1 V0
s2 V1
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
4

DOULOS

Example Design



5

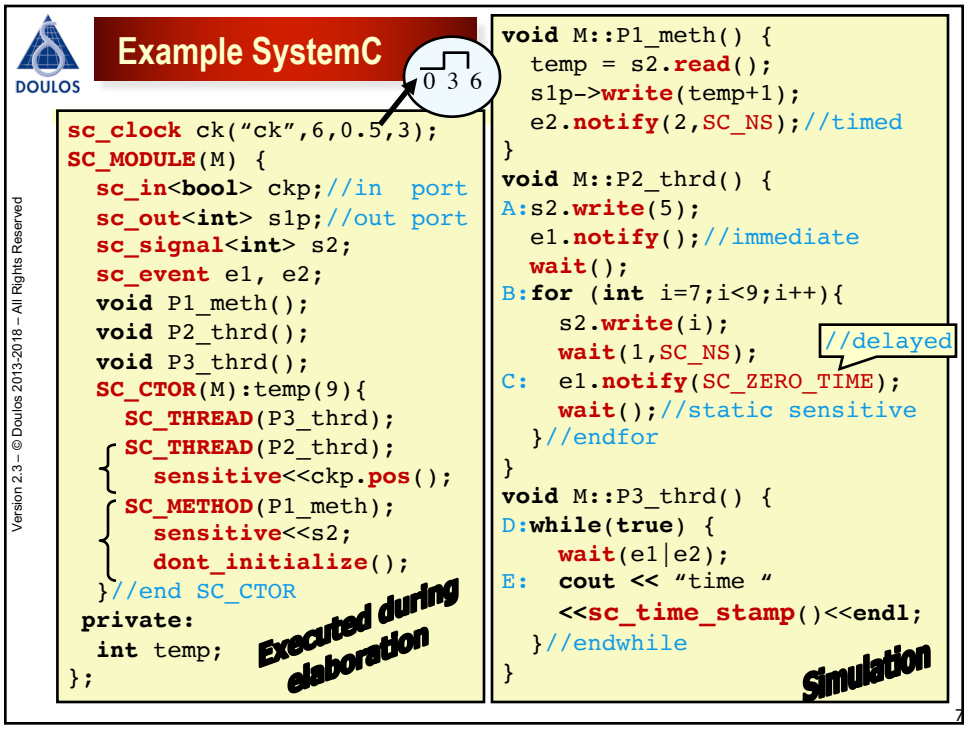
DOULOS

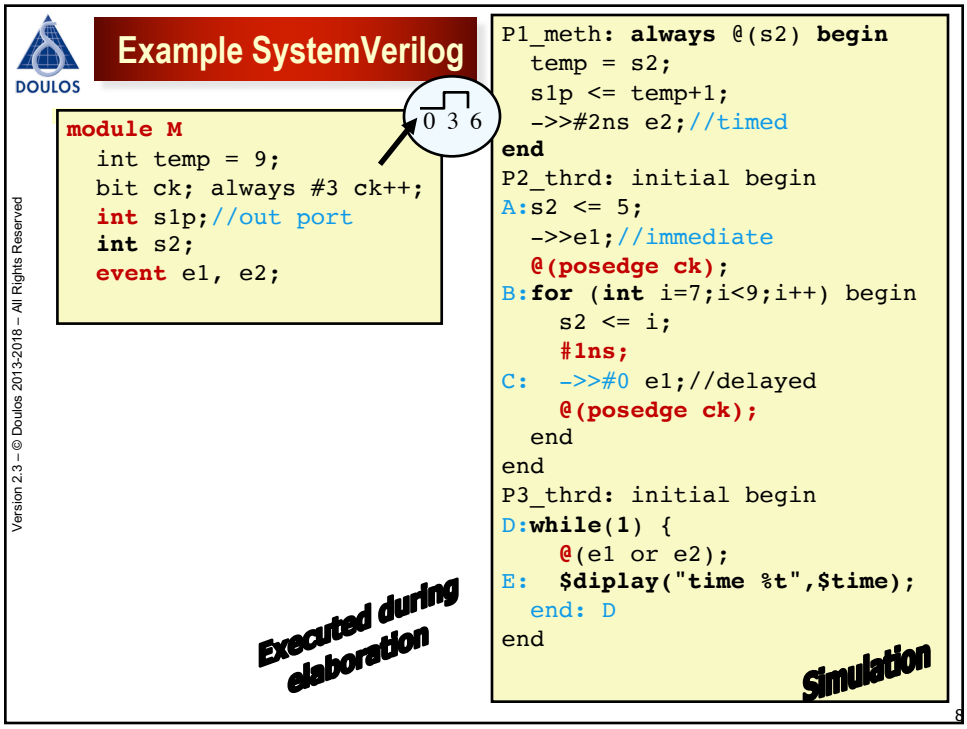
SystemC constructs used herein

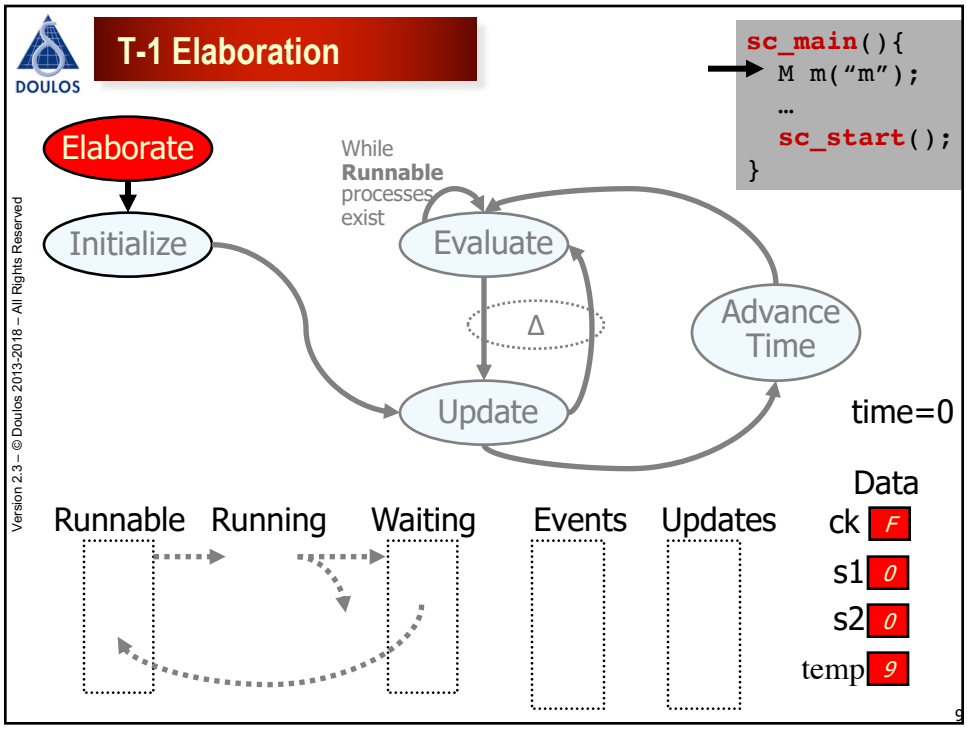
- `SC_MODULE`, `SC_CTOR` - used to create modules
- `SC_THREAD`[1], `SC_METHOD`[1] - types of processes
- `sensitive`, `dont_initialize` - attributes of processes
- `sc_event`, `wait`[2], `notify`[3] - synchronization mechanisms
- `sc_signal`[4], `read`, `write` - primitive channel
- `sc_clock` - `sc_signal`<**bool**> with a generating process
- `sc_in` - `sc_port`<> specialization of type `sc_signal_in_if`<>
- `sc_out` - `sc_port`<> specialization of type `sc_signal_out_if`<>

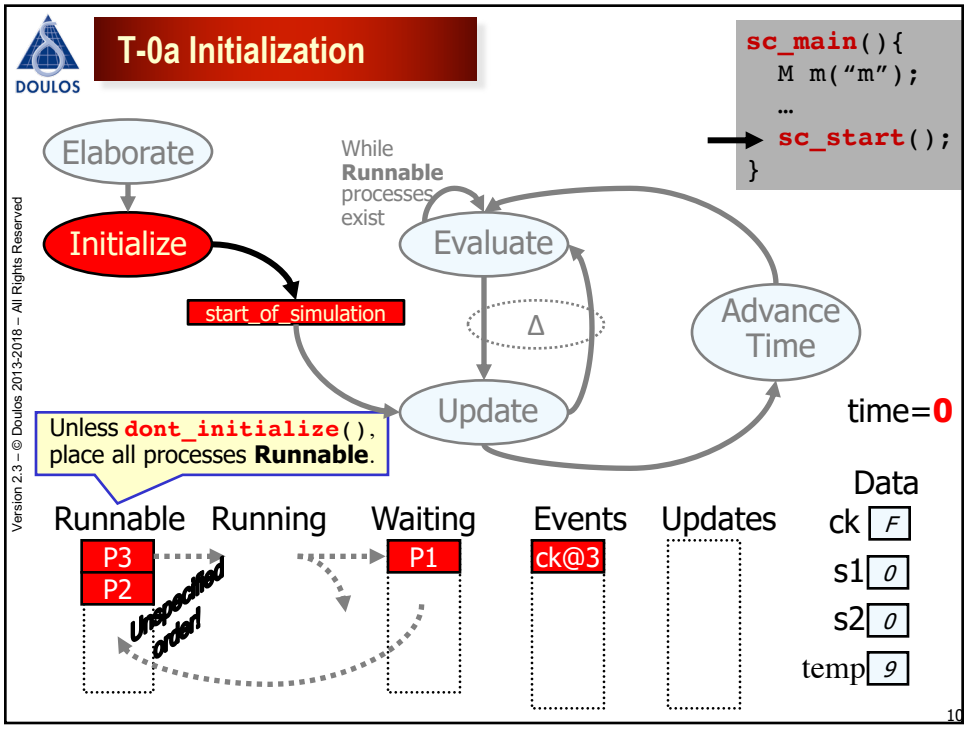
[1] Verilog **`initial`** or **`always`** block; VHDL **`process`** block
[2] Verilog **`@`**, **`#`**, or **`wait`** statement; VHDL **`wait`** statement
[3] Verilog **`->`** statement, except SystemC is more flexible
[4] Verilog **`wire`** or **`var`** with **`<=`** type; VHDL **`signal`** type

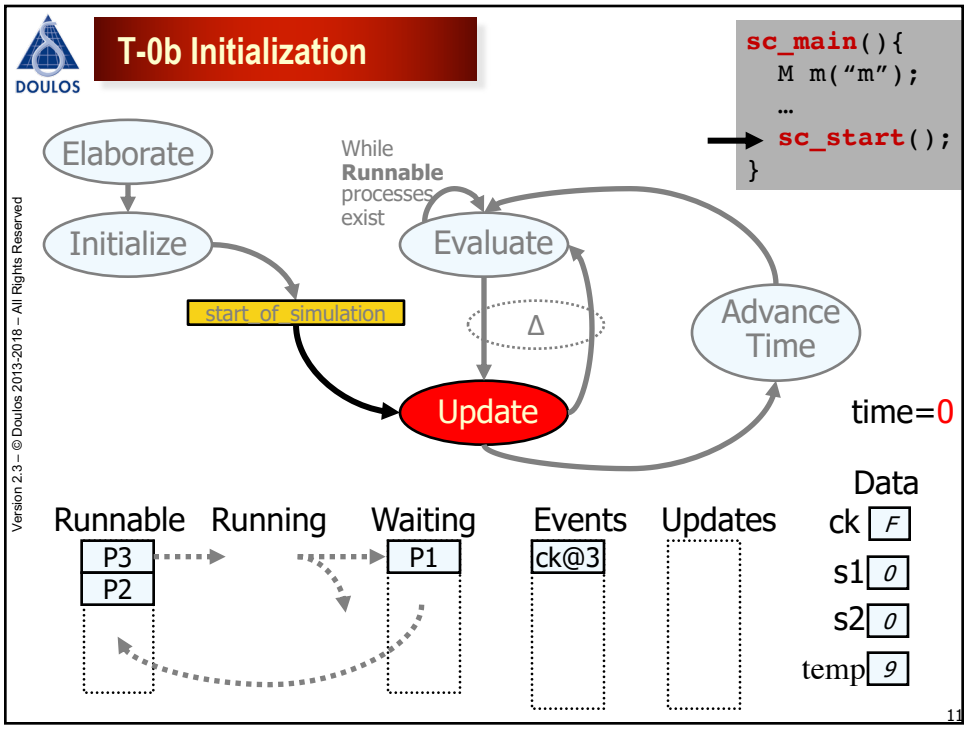
6

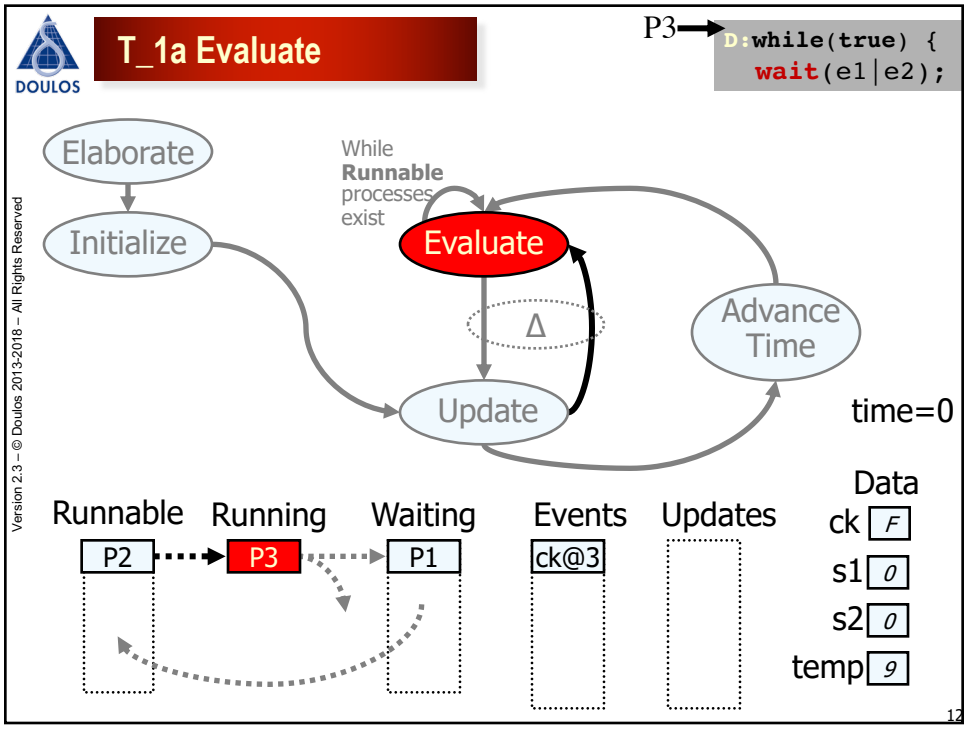

DOULOS
Example SystemC
0 3 6
sc_clock ck("ck",6,0.5,3);
SC_MODULE(M) {
sc_in<bool> ckp;//in port
sc_out<int> s1p;//out port
sc_signal<int> s2;
sc_event e1, e2;
void P1_meth();
void P2_thrd();
void P3_thrd();
SC_CTOR(M):temp(9){
SC_THREAD(P3_thrd);
SC_THREAD(P2_thrd);
sensitive<<ckp.pos();
SC_METHOD(P1_meth);
sensitive<<s2;
dont_initialize();
}//end SC_CTOR
private:
int temp;
};
Executed during elaboration
void M::P1_meth() {
temp = s2.read();
s1p->write(temp+1);
e2.notify(2,SC_NS);//timed
}
void M::P2_thrd() {
A:s2.write(5);
e1.notify();//immediate
wait();
B:for (int i=7;i<9;i++){
s2.write(i);
wait(1,SC_NS);
C: e1.notify(SC_ZERO_TIME);
wait();//static sensitive
}//endfor
}
void M::P3_thrd() {
D:while(true) {
wait(e1|e2);
E: cout << "time "
<<sc_time_stamp()<<endl;
}//endwhile
}
//delayed
Simulation
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
7

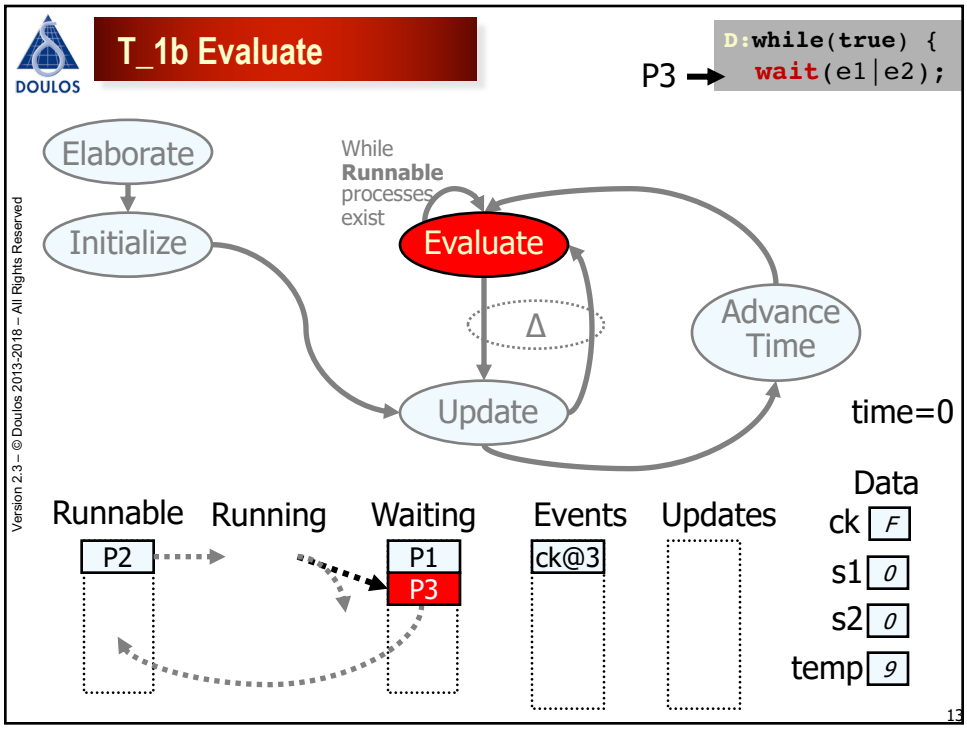

DOULOS
Example SystemVerilog
0 3 6
module M
int temp = 9;
bit ck; always #3 ck++;
int s1p;//out port
int s2;
event e1, e2;
Executed during elaboration
P1_meth: always @(s2) begin
temp = s2;
s1p <= temp+1;
->>#2ns e2;//timed
end
P2_thrd: initial begin
A:s2 <= 5;
->>e1;//immediate
@(posedge ck);
B:for (int i=7;i<9;i++) begin
s2 <= i;
#1ns;
C: ->>#0 e1;//delayed
@(posedge ck);
end
end
P3_thrd: initial begin
D:while(1) {
@(e1 or e2);
E: $diplay("time %t",$time);
end: D
end
Simulation
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
8

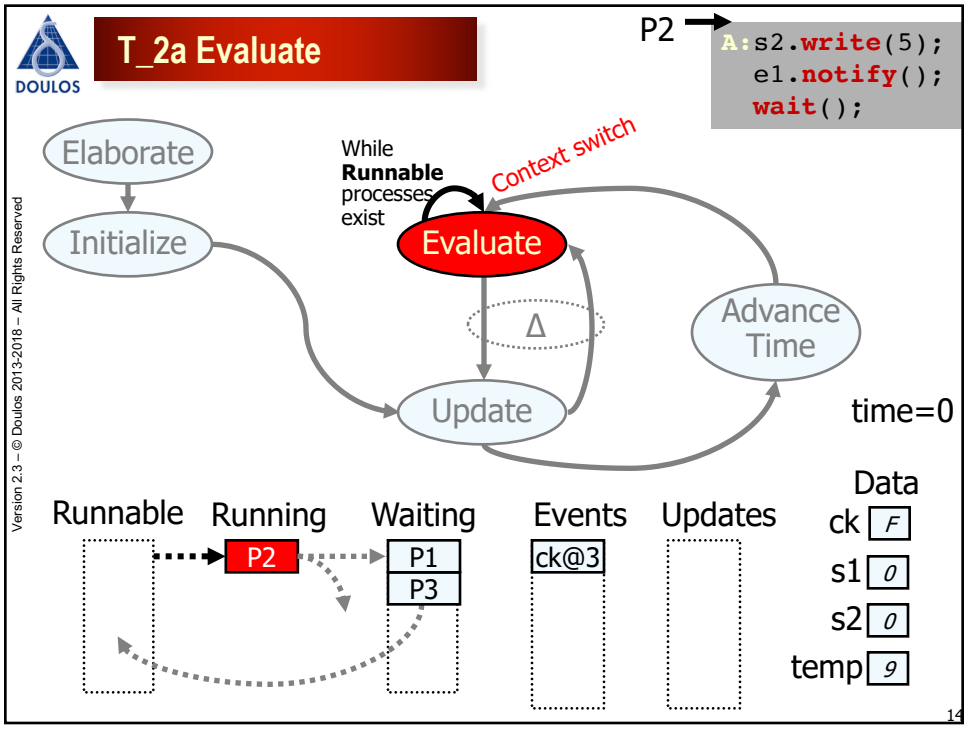

DOULOS
T-1 Elaboration
sc_main(){
M m("m");
…
sc_start();
}
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Data
Runnable
Running
Waiting
Events
Updates
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
9

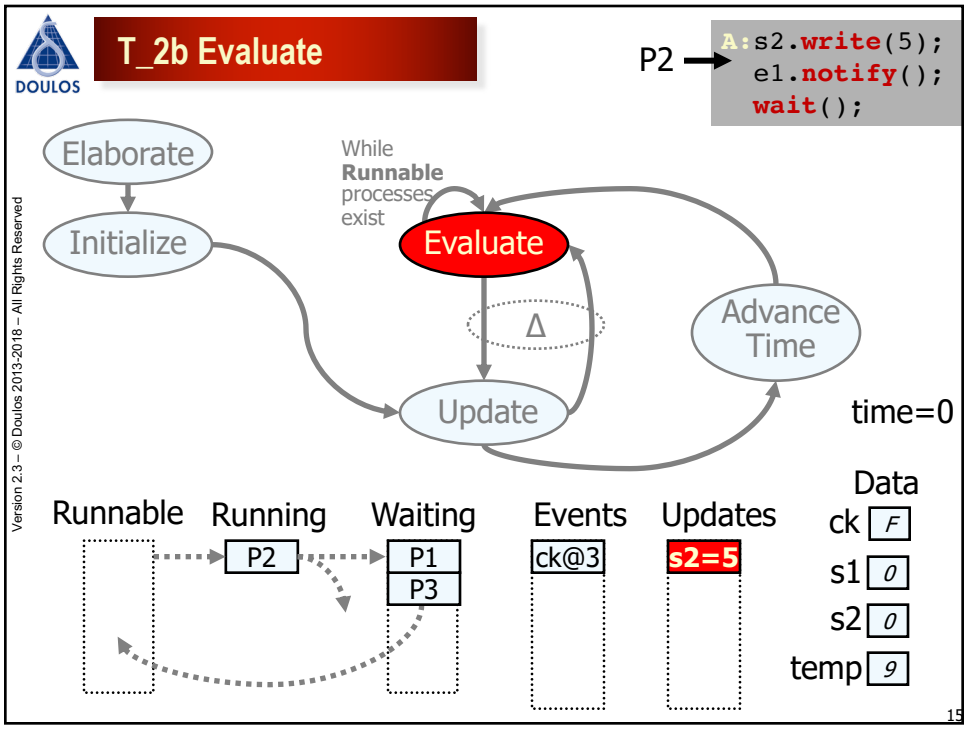

DOULOS
T-0a Initialization
sc_main(){
M m("m");
…
sc_start();
}
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
start_of_simulation
Δ
Update
Advance
Time
time=0
Unless dont_initialize(),
place all processes Runnable.
Data
Runnable
Running
Waiting
Events
Updates
P3
P2
Unspecified order!
P1
ck@3
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
10

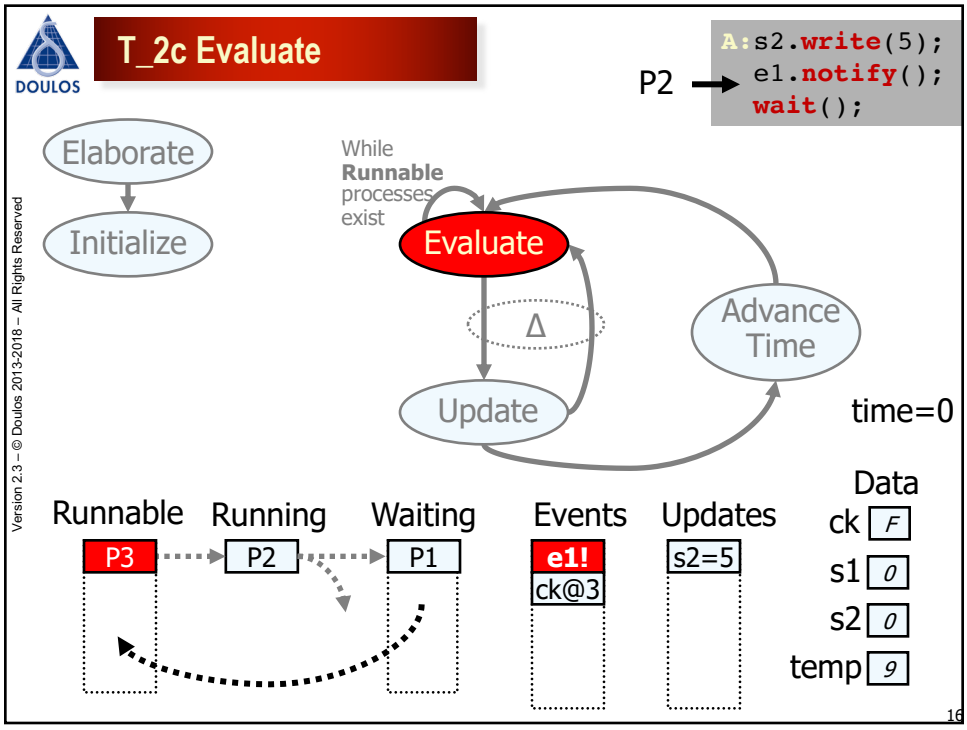

DOULOS
T-0b Initialization
sc_main(){
M m("m");
…
sc_start();
}
Elaborate
Initialize
start_of_simulation
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P3
P2
P1
ck@3
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
11

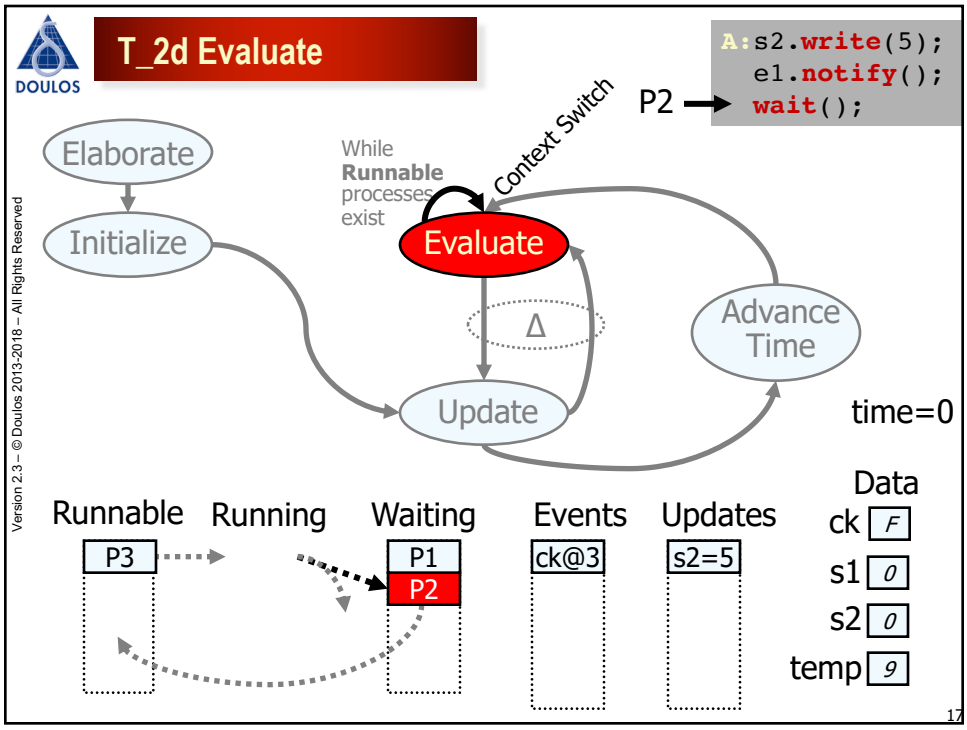

DOULOS
T_1a Evaluate
P3
D:while(true) {
wait(e1|e2);
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P2
P3
P1
ck@3
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
12

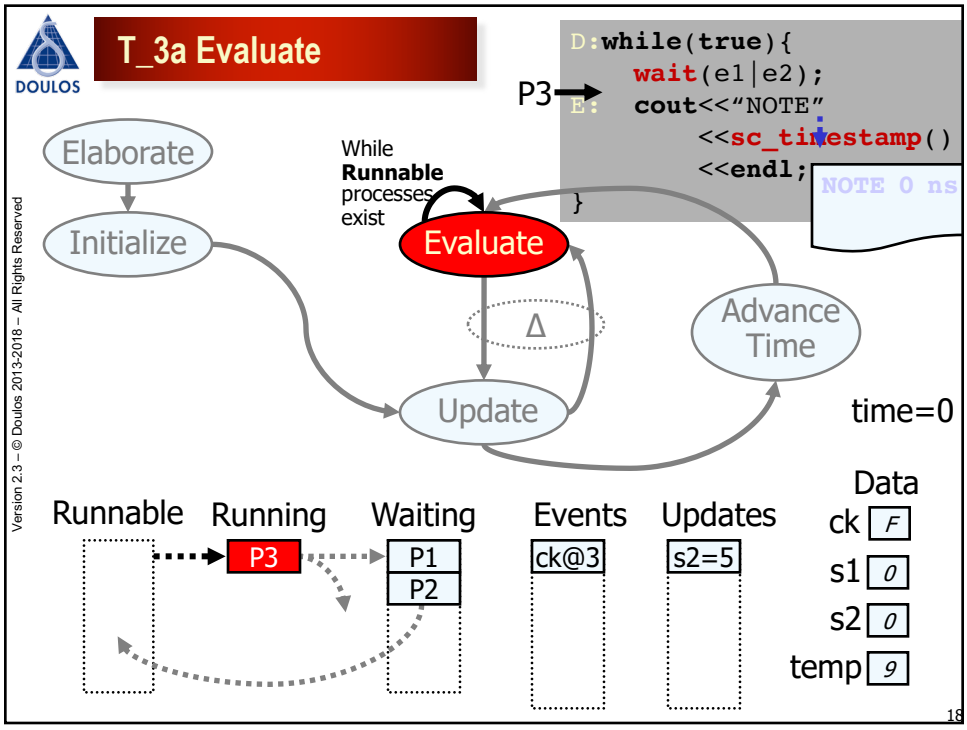

DOULOS
T_1b Evaluate
P3
D:while(true) {
wait(e1|e2);
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P2
P1
P3
ck@3
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
13

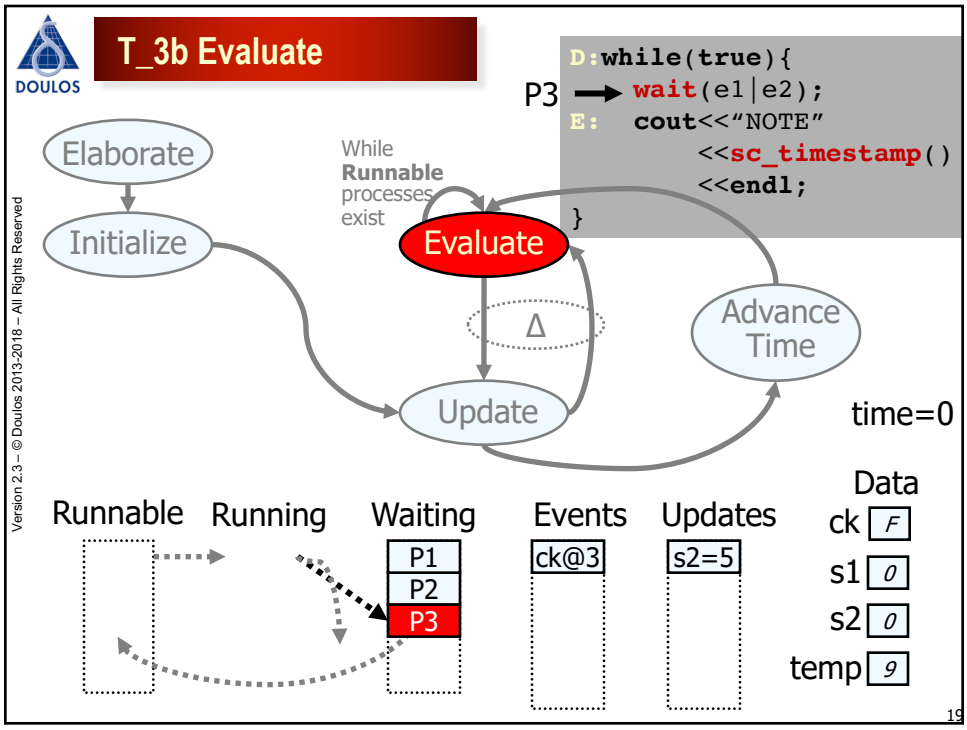

DOULOS
T_2a Evaluate
P2
A:s2.write(5);
e1.notify();
wait();
Elaborate
Initialize
While
Runnable
processes
exist
Context switch
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P2
P1
P3
ck@3
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
14

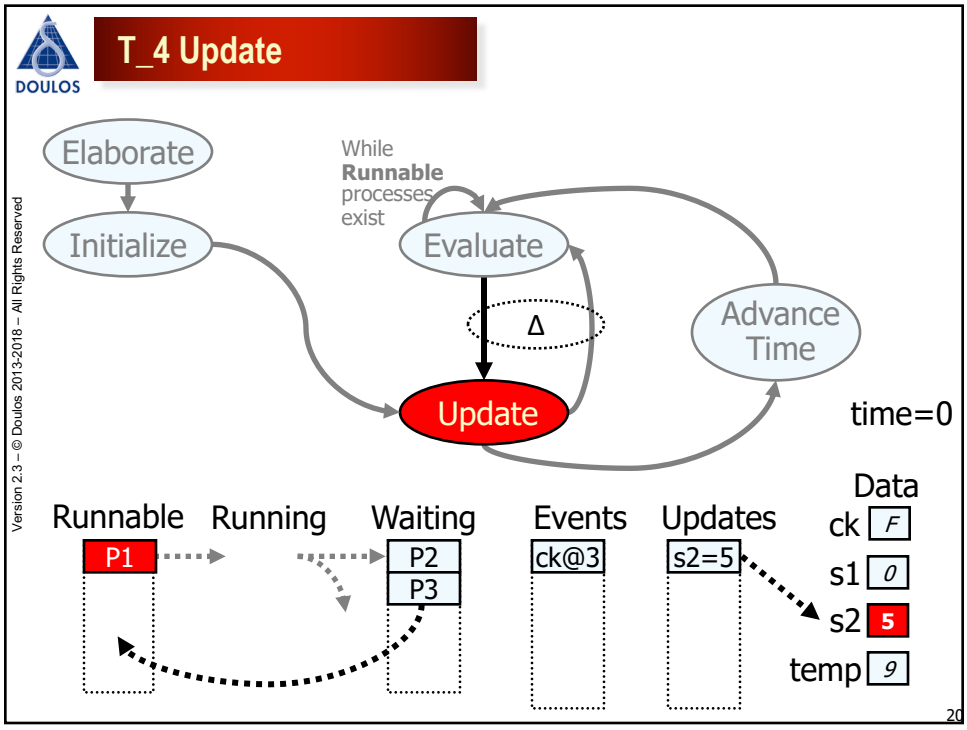

DOULOS
T_2b Evaluate
P2
A: s2.write(5);
e1.notify();
wait();
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P2
P1
P3
ck@3
s2=5
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
15

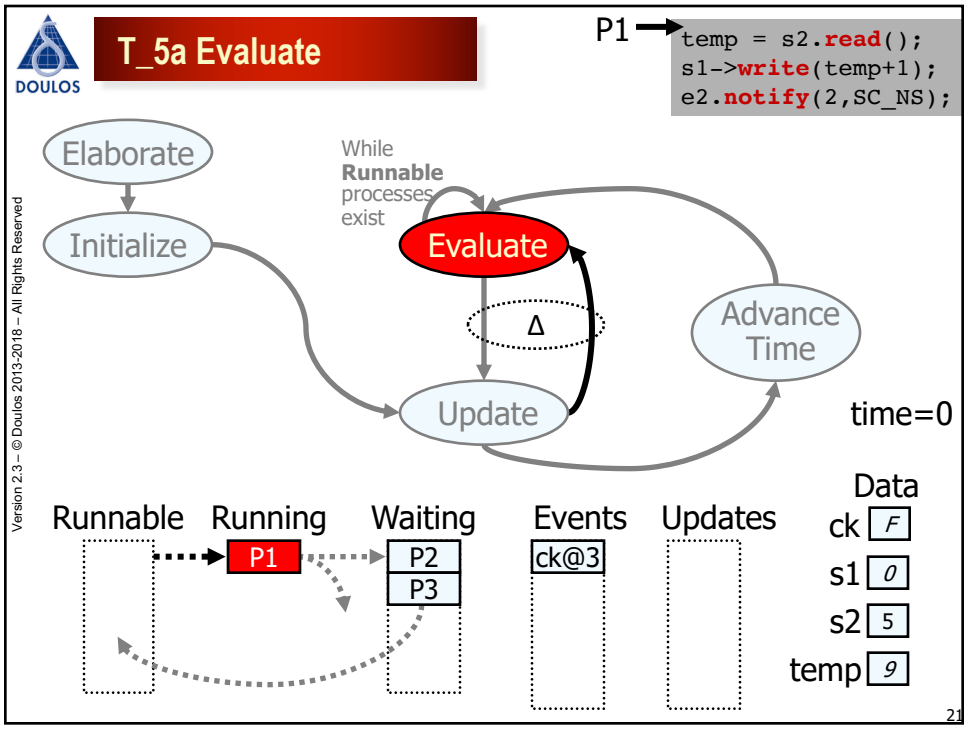

DOULOS
T_2c Evaluate
P2
A: s2.write(5);
e1.notify();
wait();
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P3
P2
P1
e1!
ck@3
s2=5
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
16

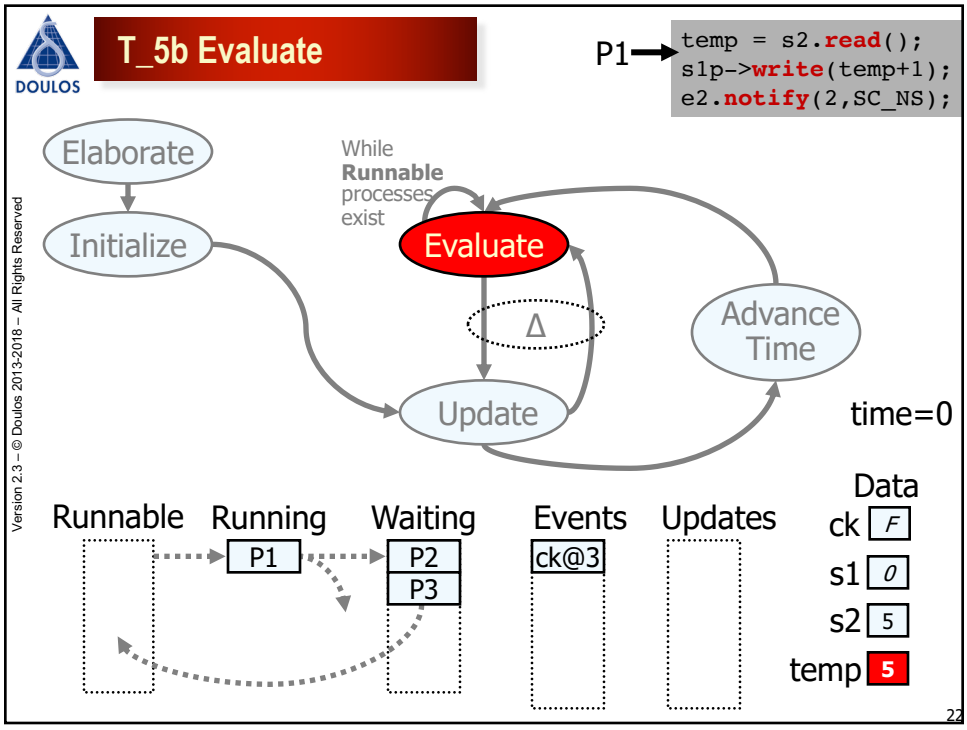

DOULOS
T_2d Evaluate
A:s2.write(5);
e1.notify();
wait();
P2
Context Switch
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P3
P1
P2
ck@3
s2=5
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
17

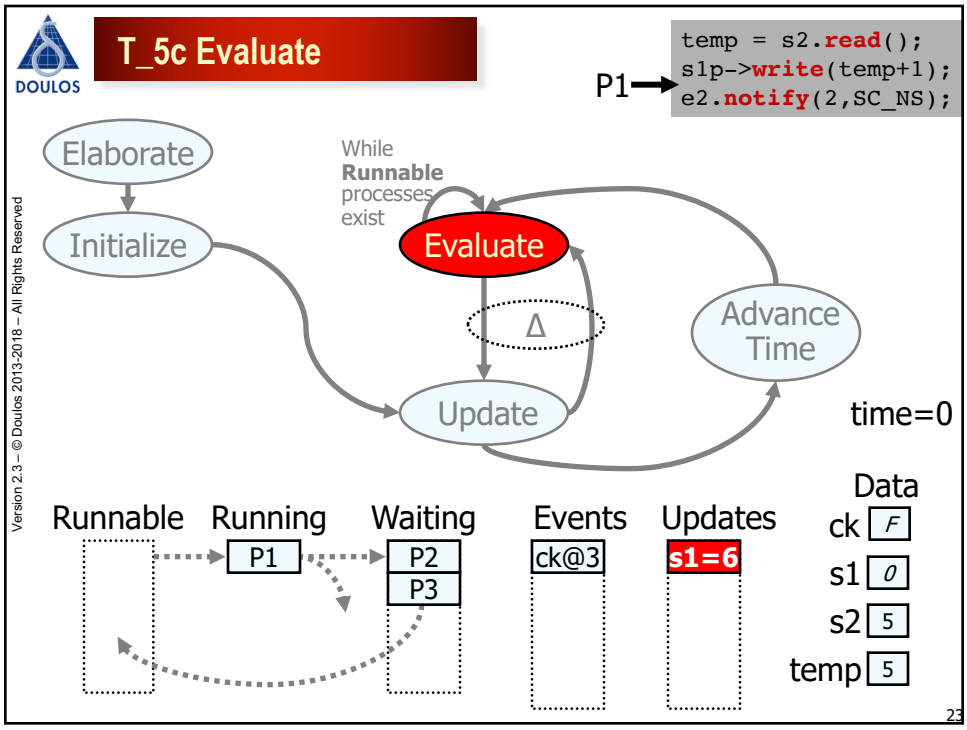

DOULOS
T_3a Evaluate
D:while(true){
wait(e1|e2);
E: cout<<"NOTE"
<<sc_timestamp()
<<endl;
}
NOTE 0 ns
P3
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P3
P1
P2
ck@3
s2=5
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
18

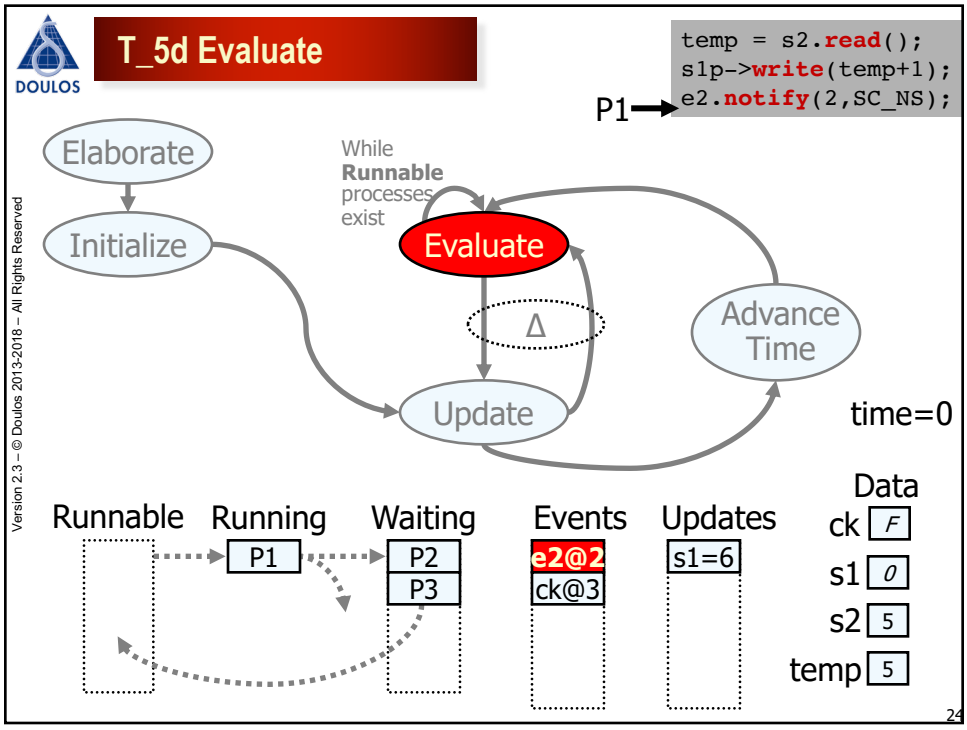

DOULOS
T_3b Evaluate
D:while(true){
P3 wait(e1|e2);
E: cout<<"NOTE"
<<sc_timestamp()
<<endl;
}
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P1
P2
P3
ck@3
s2=5
ck F
s1 0
s2 0
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
19

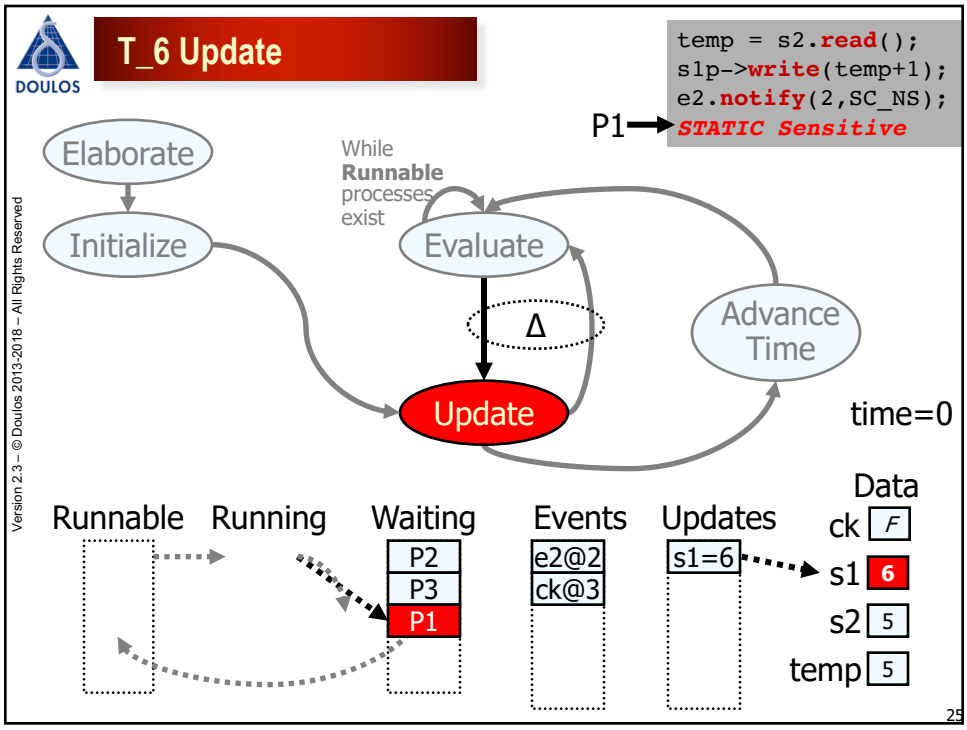

DOULOS
T_4 Update
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P1
P2
P3
ck@3
s2=5
ck F
s1 0
s2 5
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
20

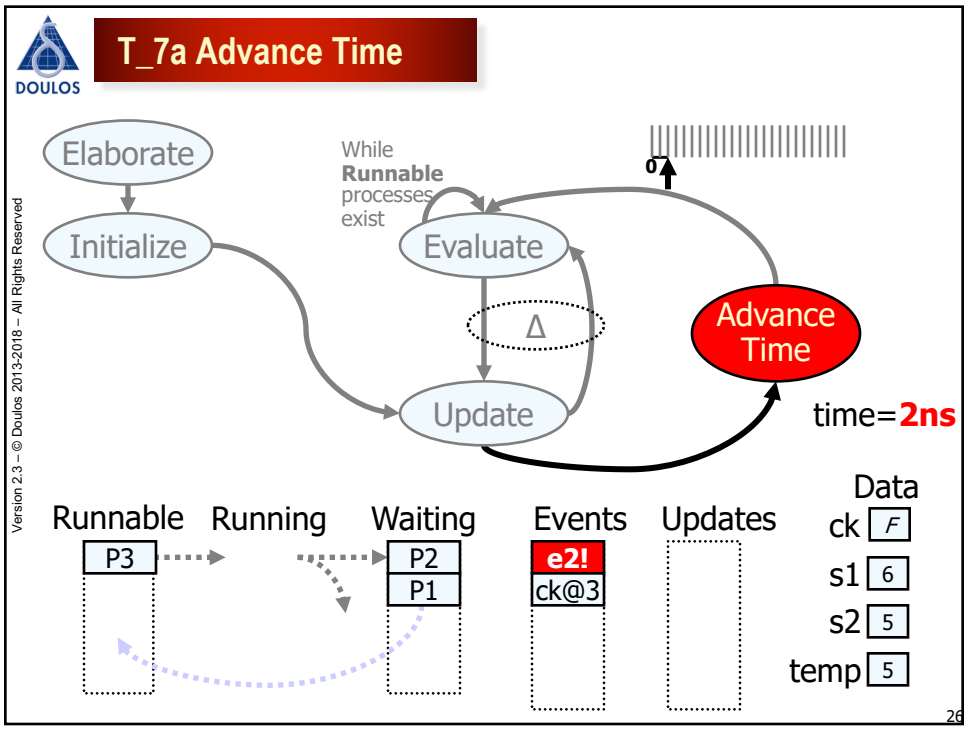

DOULOS
T_5a Evaluate
P1
temp = s2.read();
s1->write(temp+1);
e2.notify(2,SC_NS);
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
P1
P2
P3
ck@3
Data
ck F
s1 0
s2 5
temp 9
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
21

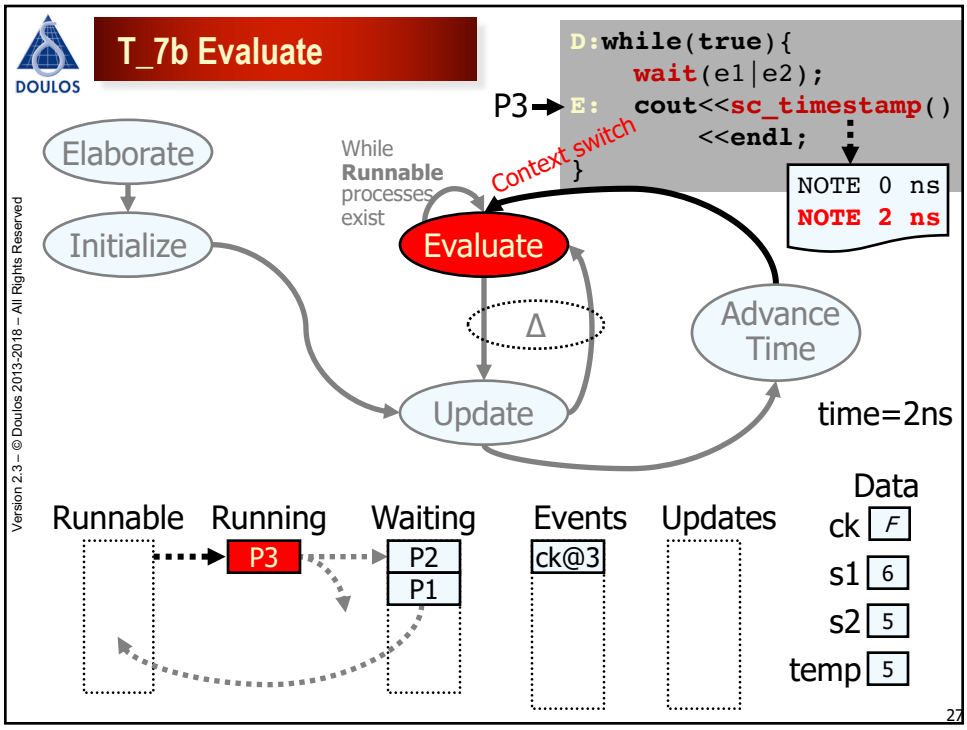

DOULOS
T_5b Evaluate
P1
temp = s2.read();
s1p->write(temp+1);
e2.notify(2,SC_NS);
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
P1
P2
P3
ck@3
Data
ck F
s1 0
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
22

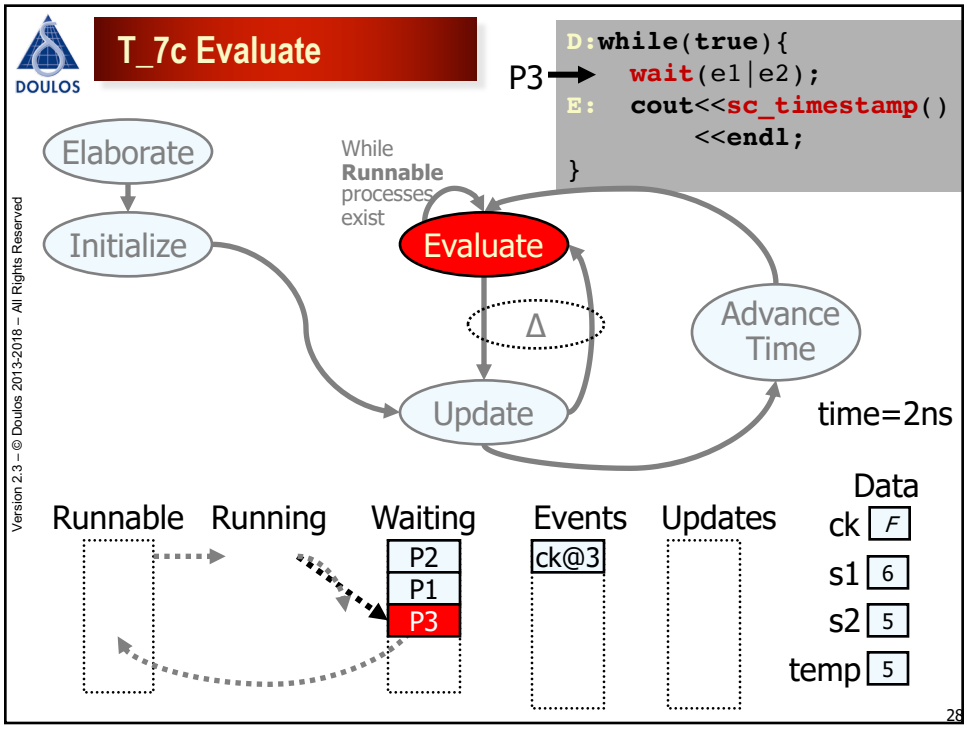

DOULOS
T_5c Evaluate
temp = s2.read();
s1p->write(temp+1);
e2.notify(2,SC_NS);
P1
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P1
P2
P3
ck@3
s1=6
ck F
s1 0
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
23

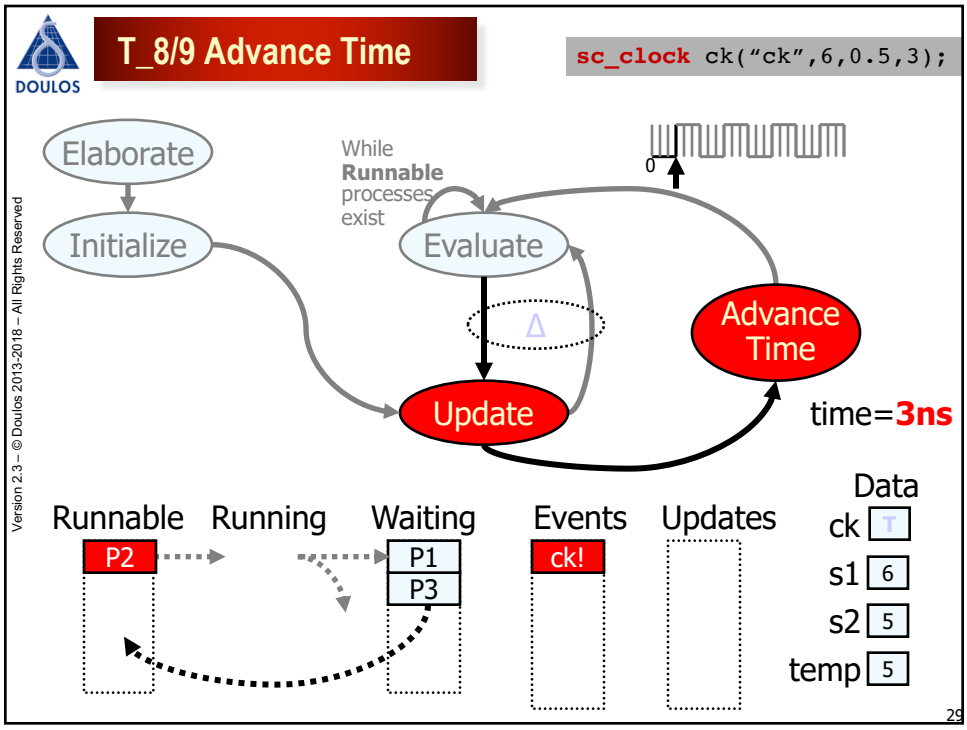

DOULOS
T_5d Evaluate
temp = s2.read();
s1p->write(temp+1);
e2.notify(2,SC_NS);
P1
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P1
P2
P3
e2@2
ck@3
s1=6
ck F
s1 0
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
24

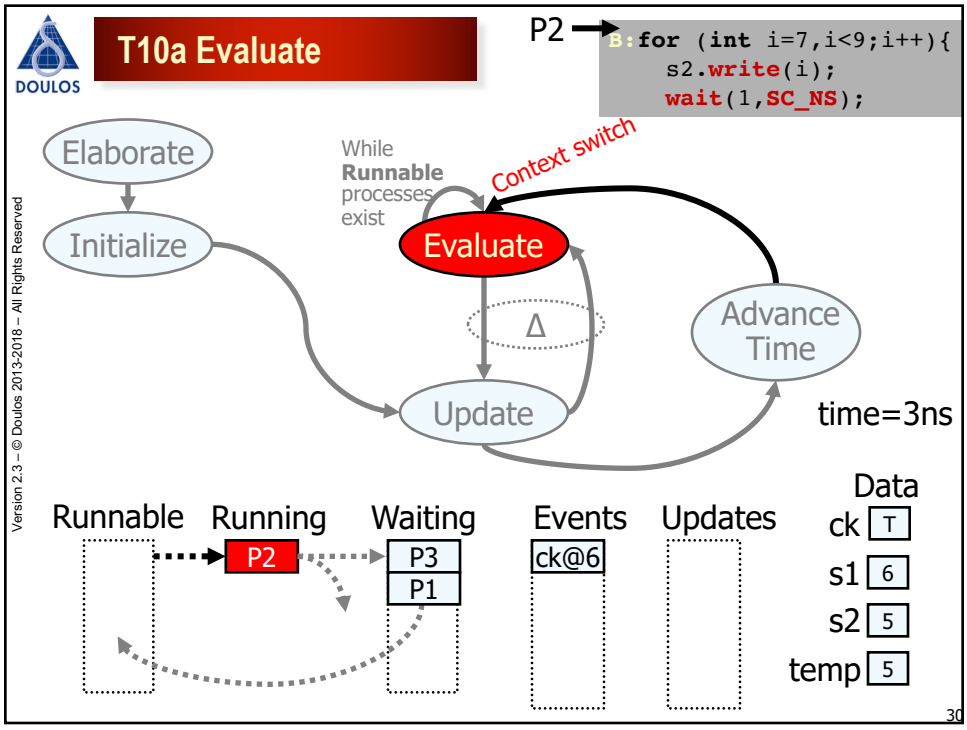

DOULOS
T_6 Update
temp = s2.read();
s1p->write(temp+1);
e2.notify(2,SC_NS);
STATIC Sensitive
P1
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=0
Runnable
Running
Waiting
Events
Updates
Data
P2
P3
P1
e2@2
ck@3
s1=6
ck F
s1 6
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
25

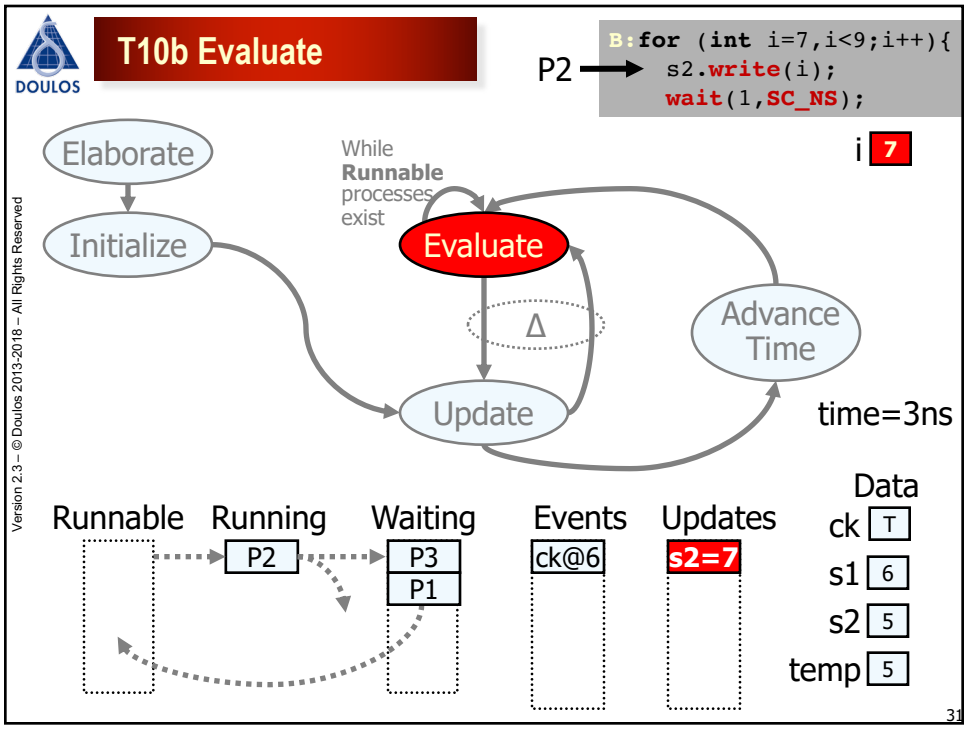

DOULOS
T_7a Advance Time
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
0
Advance
Time
time=2ns
Runnable
Running
Waiting
Events
Updates
Data
P3
P2
P1
e2!
ck@3
ck F
s1 6
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
26

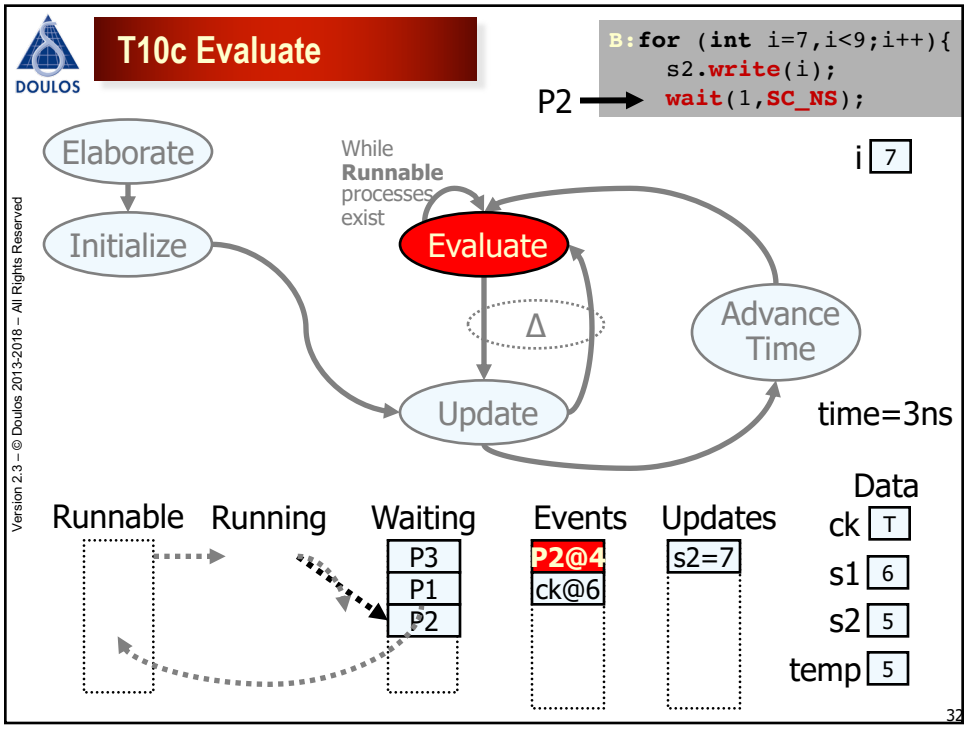

DOULOS
T_7b Evaluate
D:while(true){
wait(e1|e2);
E: cout<<sc_timestamp()
<<endl;
}
P3
Context switch
NOTE 0 ns
NOTE 2 ns
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=2ns
Runnable
Running
Waiting
Events
Updates
Data
P3
P2
P1
ck@3
ck F
s1 6
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
27

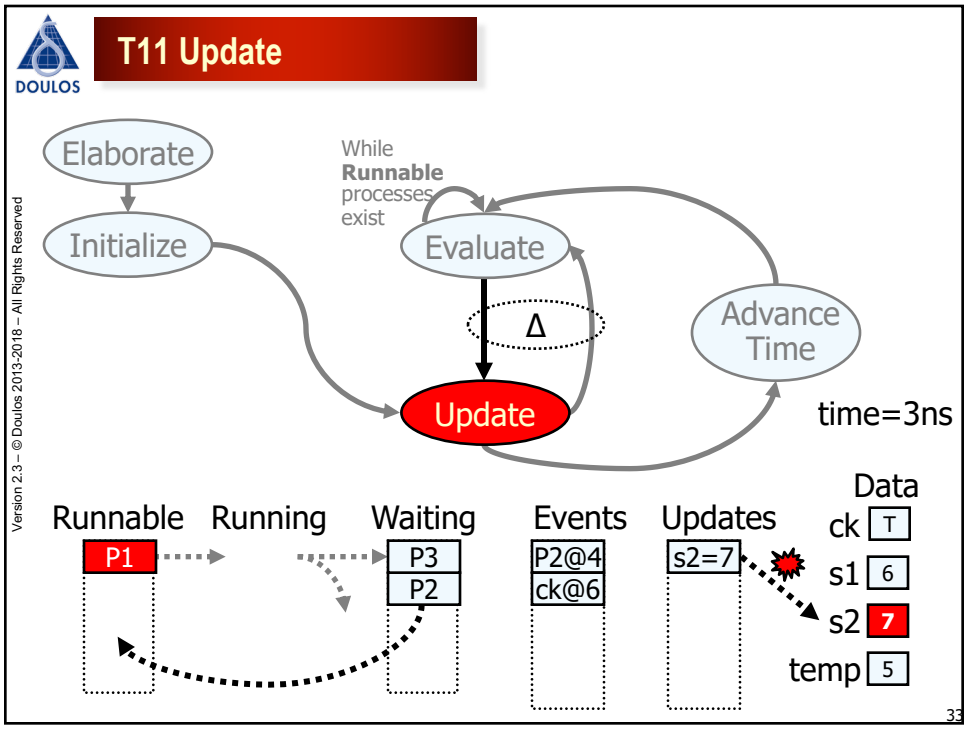

DOULOS
T_7c Evaluate
D:while(true){
wait(e1|e2);
E: cout<<sc_timestamp()
<<endl;
}
P3
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=2ns
Runnable
Running
Waiting
Events
Updates
Data
P2
P1
P3
ck@3
ck F
s1 6
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
28

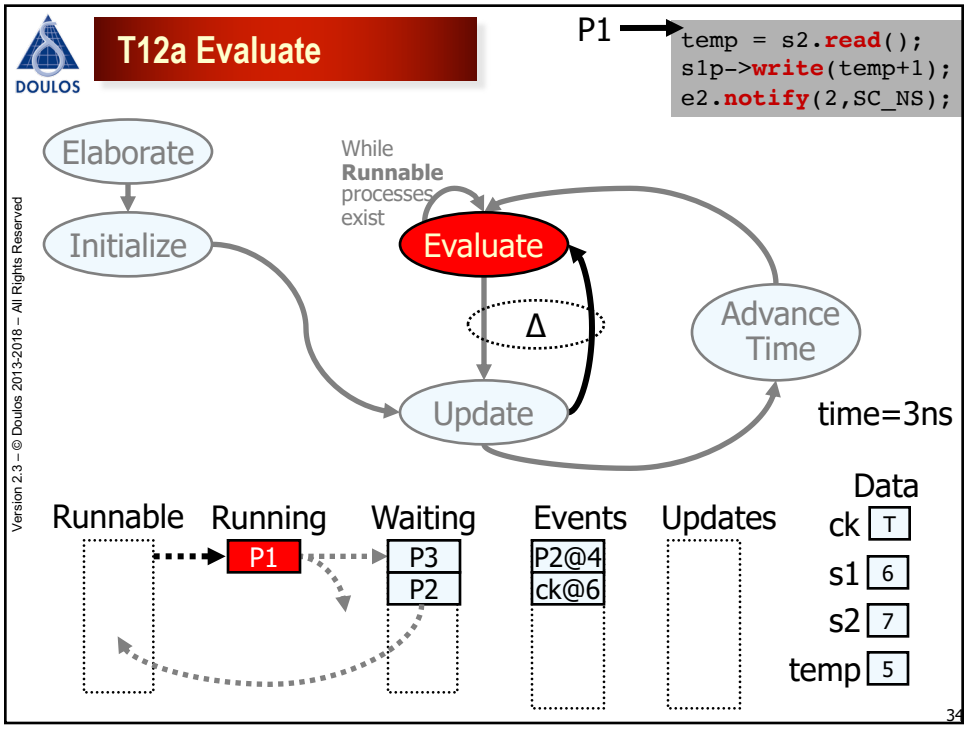

DOULOS
T_8/9 Advance Time
sc_clock ck("ck",6,0.5,3);
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
0
time=3ns
Runnable
Running
Waiting
Events
Updates
Data
P2
P1
P3
ck!
ck T
s1 6
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
29

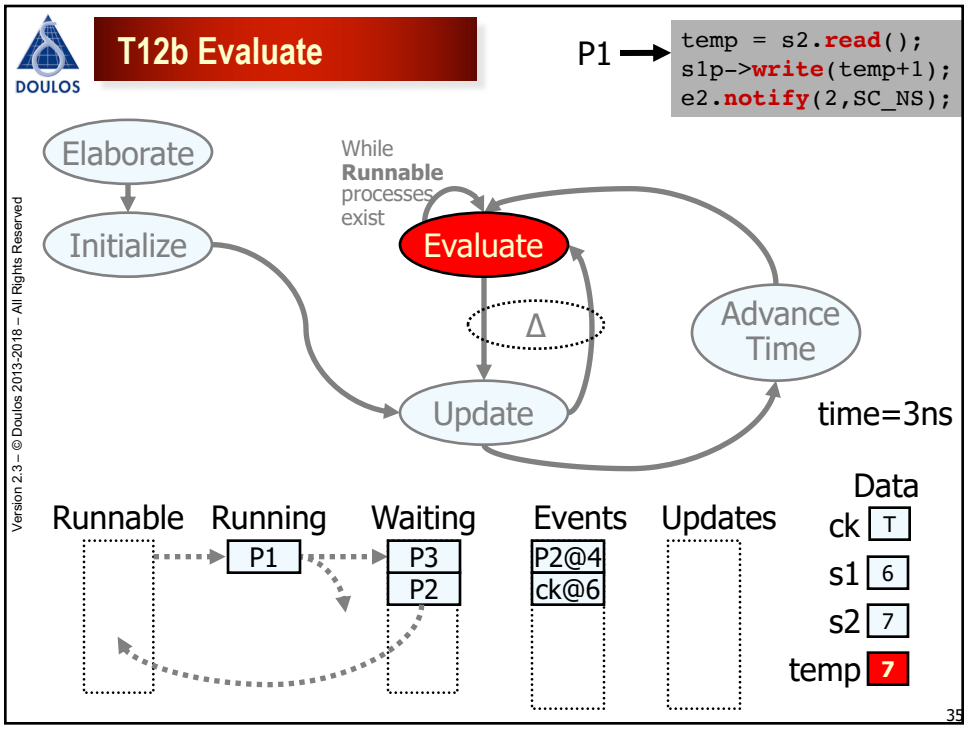

DOULOS
T10a Evaluate
P2
B:for (int i=7,i<9;i++){
s2.write(i);
wait(1,SC_NS);
Elaborate
Initialize
While
Runnable
processes
exist
Context switch
Evaluate
Δ
Update
Advance
Time
time=3ns
Runnable
Running
Waiting
Events
Updates
Data
P2
P3
P1
ck@6
ck T
s1 6
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
30

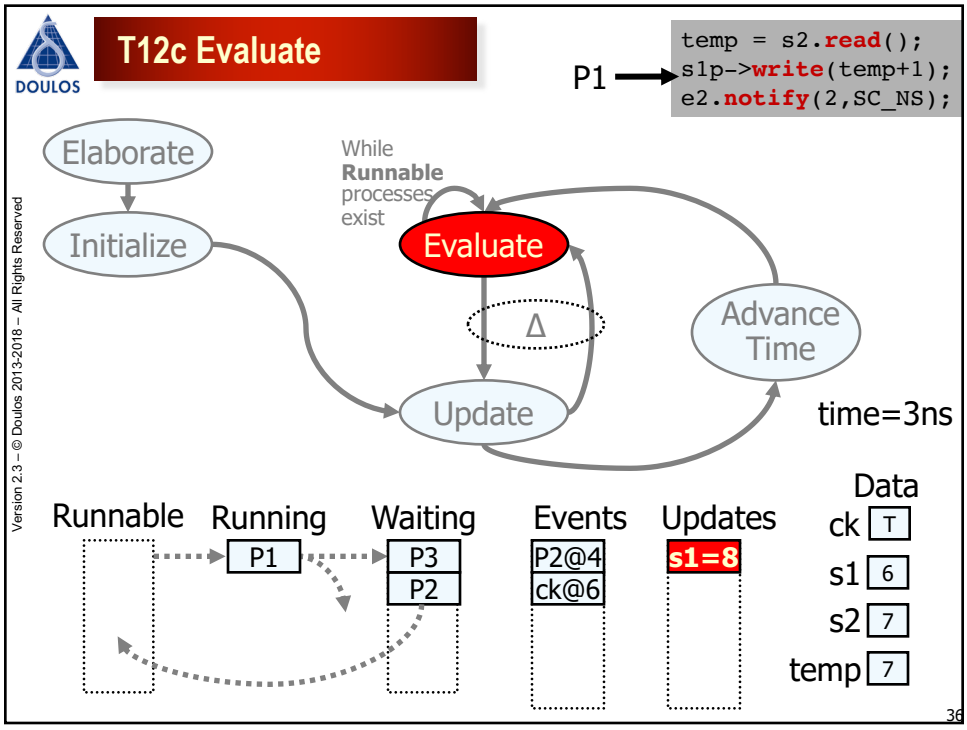

DOULOS
T10b Evaluate
P2
B:for (int i=7,i<9;i++){
s2.write(i);
wait(1,SC_NS);
i 7
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=3ns
Runnable
Running
Waiting
Events
Updates
Data
P2
P3
P1
ck@6
s2=7
ck T
s1 6
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
31

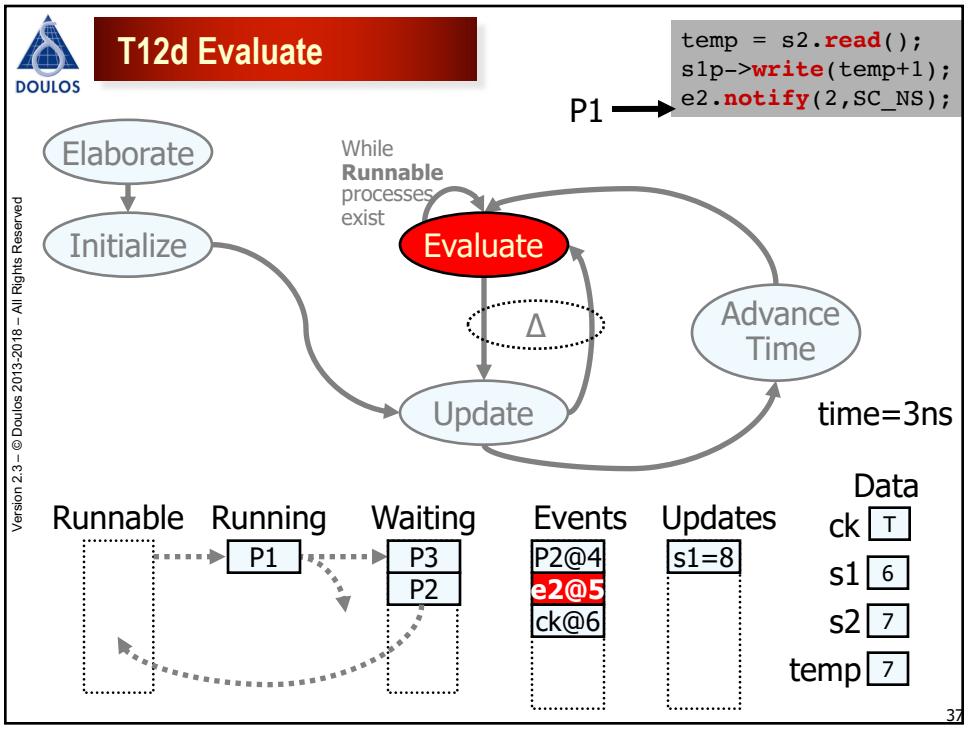

DOULOS
T10c Evaluate
P2
B:for (int i=7,i<9;i++){
s2.write(i);
wait(1,SC_NS);
i 7
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=3ns
Runnable
Running
Waiting
Events
Updates
Data
P3
P1
P2
P2@4
ck@6
s2=7
ck T
s1 6
s2 5
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
32

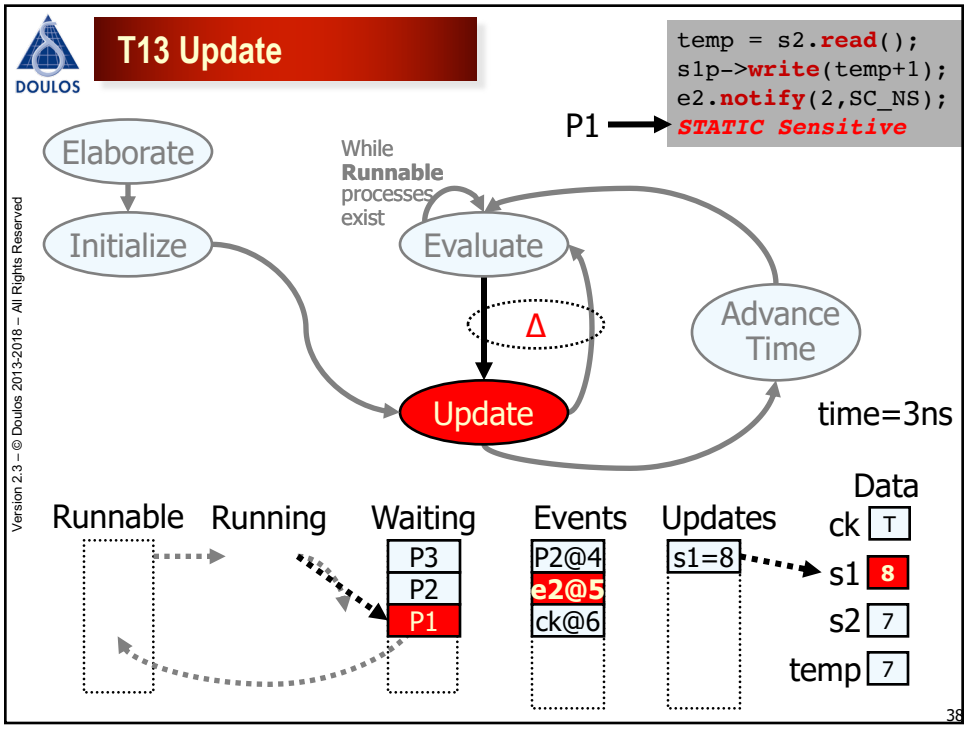

DOULOS
T11 Update
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=3ns
Runnable
Running
Waiting
Events
Updates
Data
P1
P3
P2
P2@4
ck@6
s2=7
ck T
s1 6
s2 7
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
33

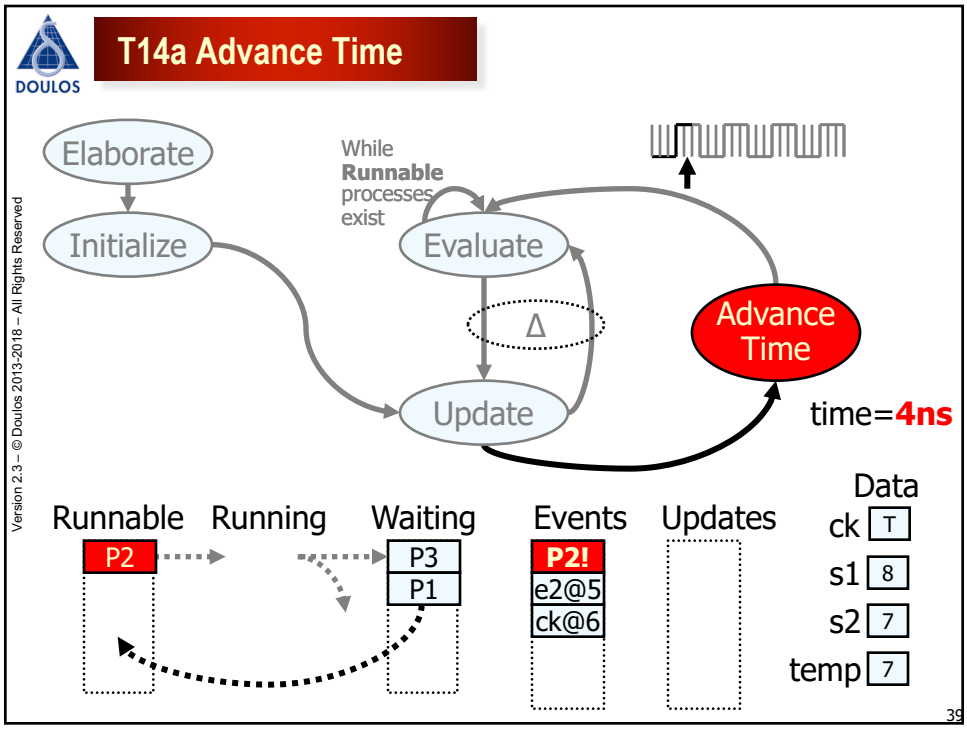

DOULOS
T12a Evaluate
P1
temp = s2.read();
s1p->write(temp+1);
e2.notify(2,SC_NS);
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=3ns
Runnable
Running
Waiting
Events
Updates
Data
P1
P3
P2
P2@4
ck@6
ck T
s1 6
s2 7
temp 5
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
34

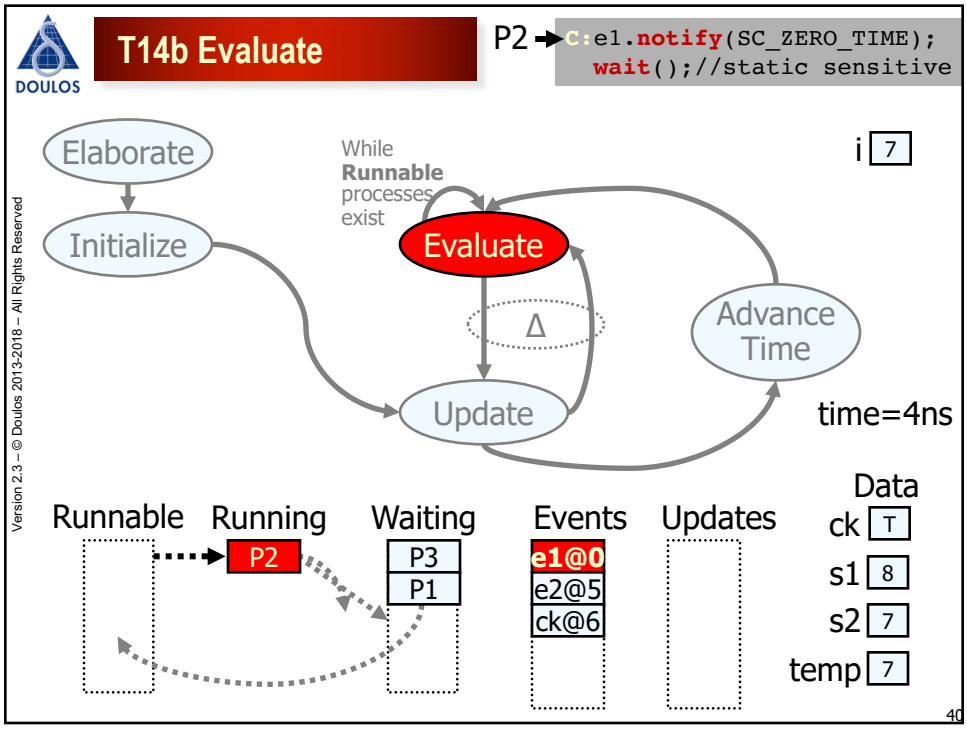

DOULOS
T12b Evaluate
P1
temp = s2.read();
s1p->write(temp+1);
e2.notify(2,SC_NS);
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=3ns
Runnable
Running
Waiting
Events
Updates
P1
P3
P2
P2@4
ck@6
Data
ck T
s1 6
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
35

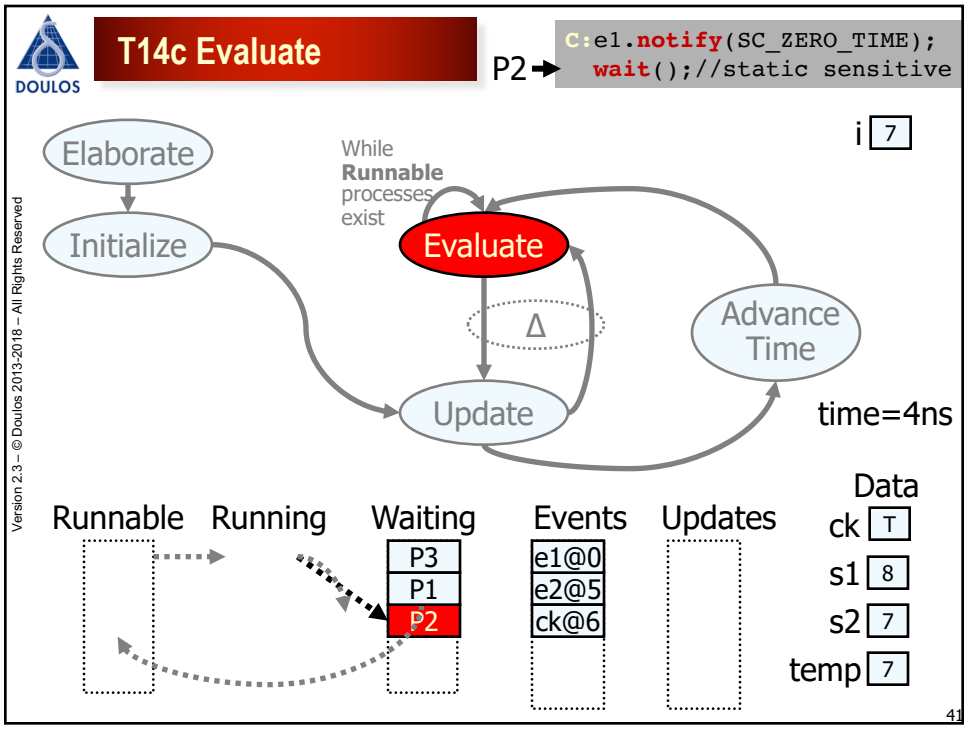

DOULOS
T12c Evaluate
P1
temp = s2.read();
s1p->write(temp+1);
e2.notify(2,SC_NS);
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=3ns
Runnable
Running
Waiting
Events
Updates
P1
P3
P2
P2@4
ck@6
s1=8
Data
ck T
s1 6
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
36

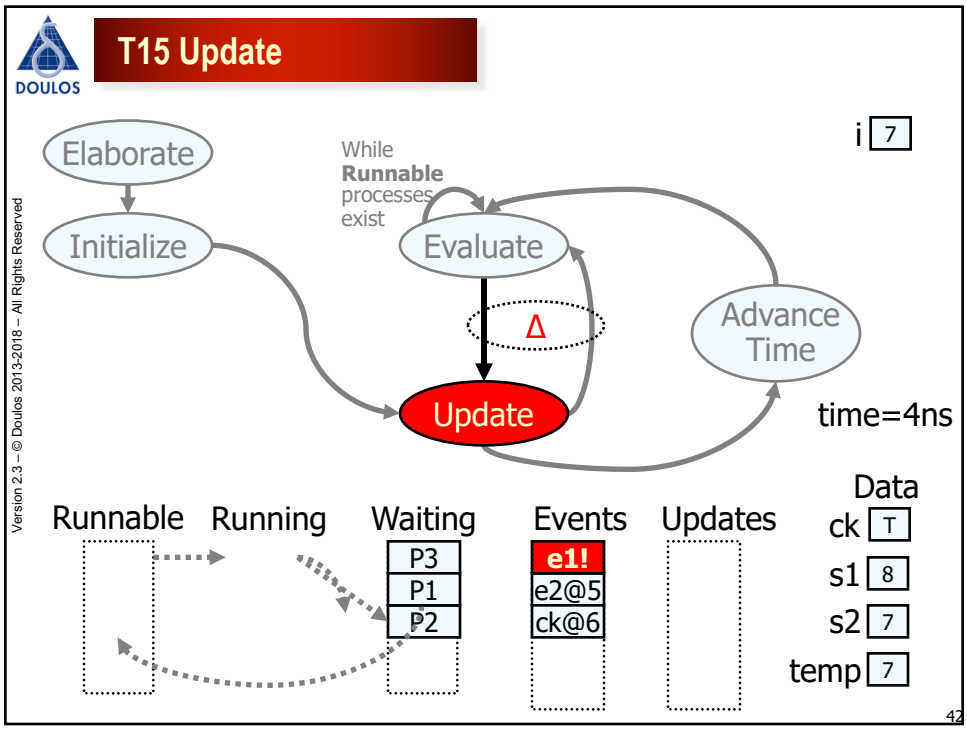

DOULOS
T12d Evaluate
temp = s2.read();
s1p->write(temp+1);
e2.notify(2,SC_NS);
P1
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=3ns
Runnable
Running
Waiting
Events
Updates
Data
P1
P3
P2
P2@4
e2@5
ck@6
s1=8
ck T
s1 6
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
37

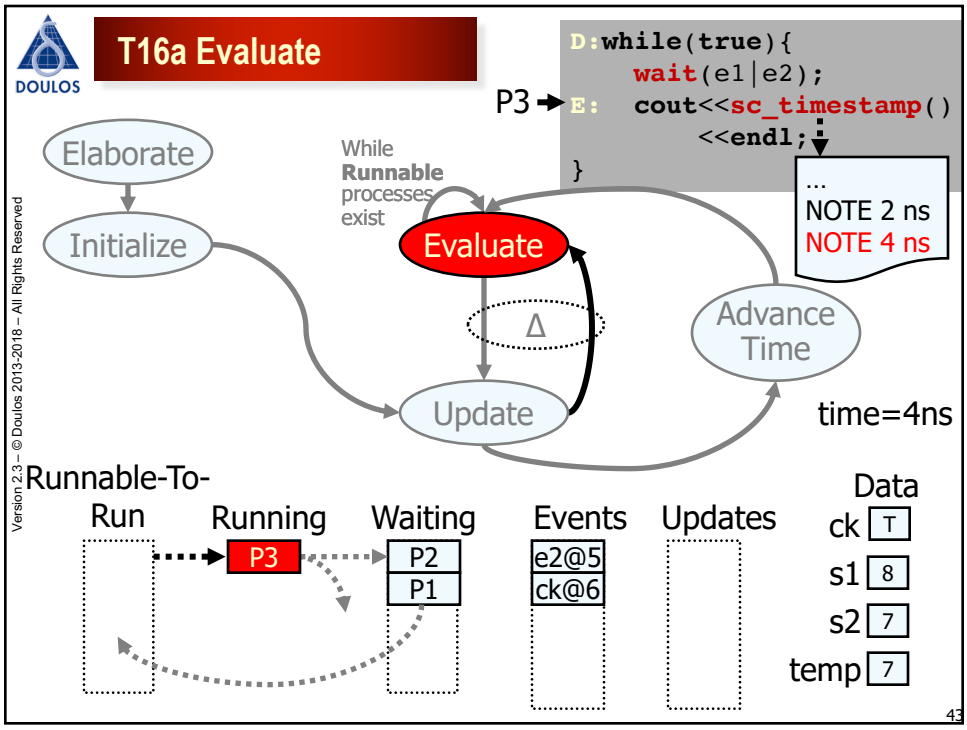

DOULOS
T13 Update
temp = s2.read();
s1p->write(temp+1);
e2.notify(2,SC_NS);
STATIC Sensitive
P1
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=3ns
Runnable
Running
Waiting
Events
Updates
Data
P3
P2
P1
P2@4
e2@5
ck@6
s1=8
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
38

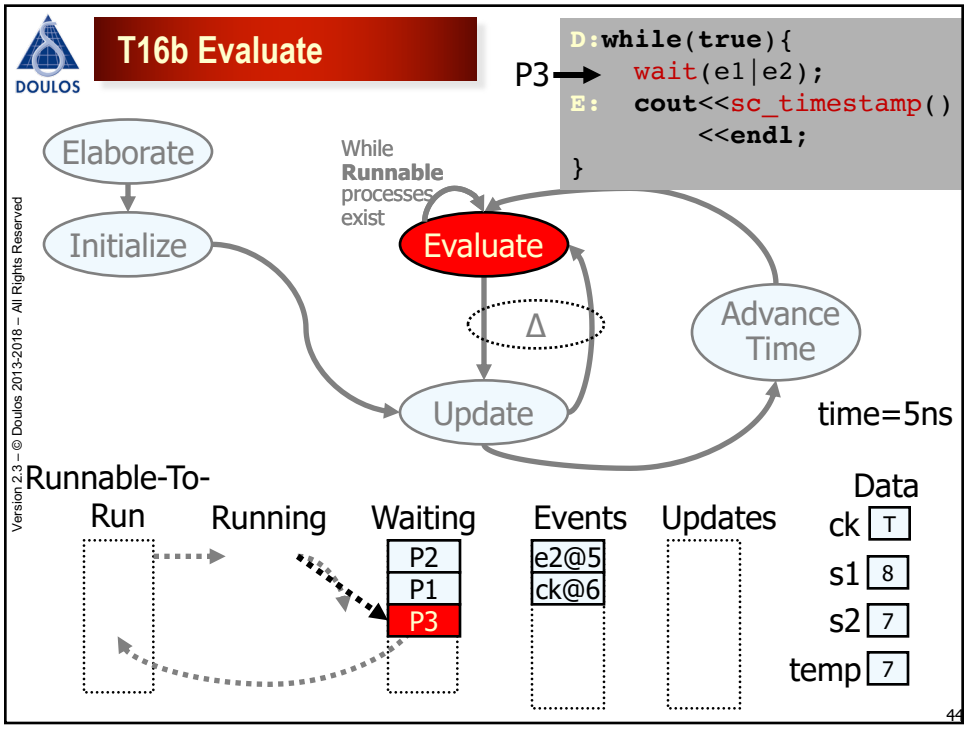

DOULOS
T14a Advance Time
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=4ns
Runnable
Running
Waiting
Events
Updates
Data
P2
P3
P1
P2!
e2@5
ck@6
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
39

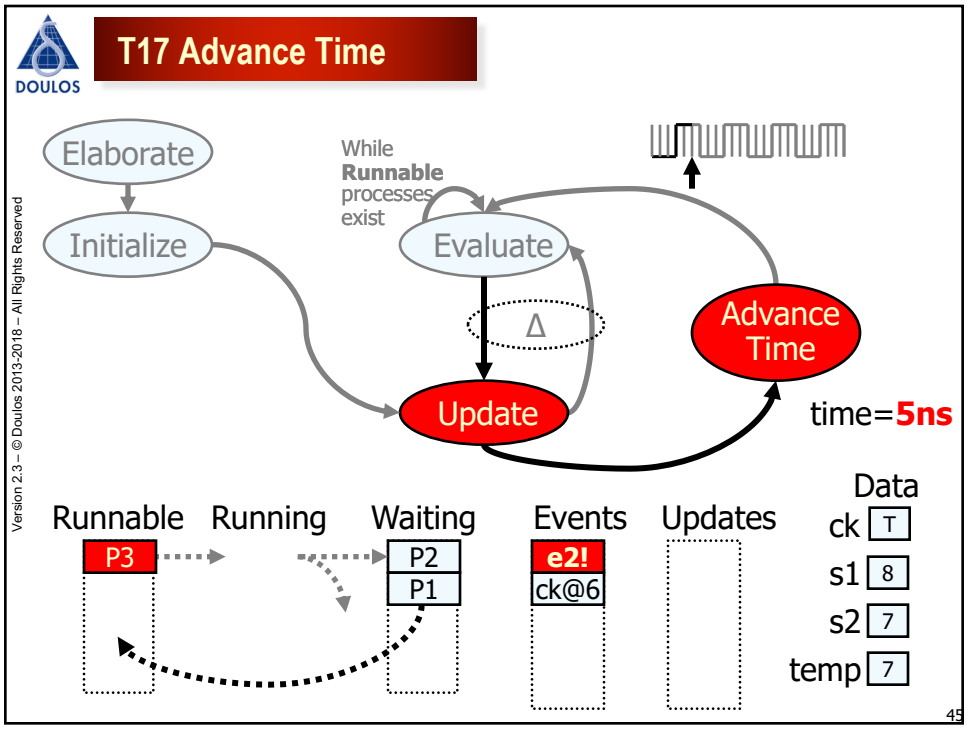

DOULOS
T14b Evaluate
P2
C:e1.notify(SC_ZERO_TIME);
wait();//static sensitive
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=4ns
i 7
Runnable
Running
Waiting
Events
Updates
Data
P2
P3
P1
e1@0
e2@5
ck@6
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
40

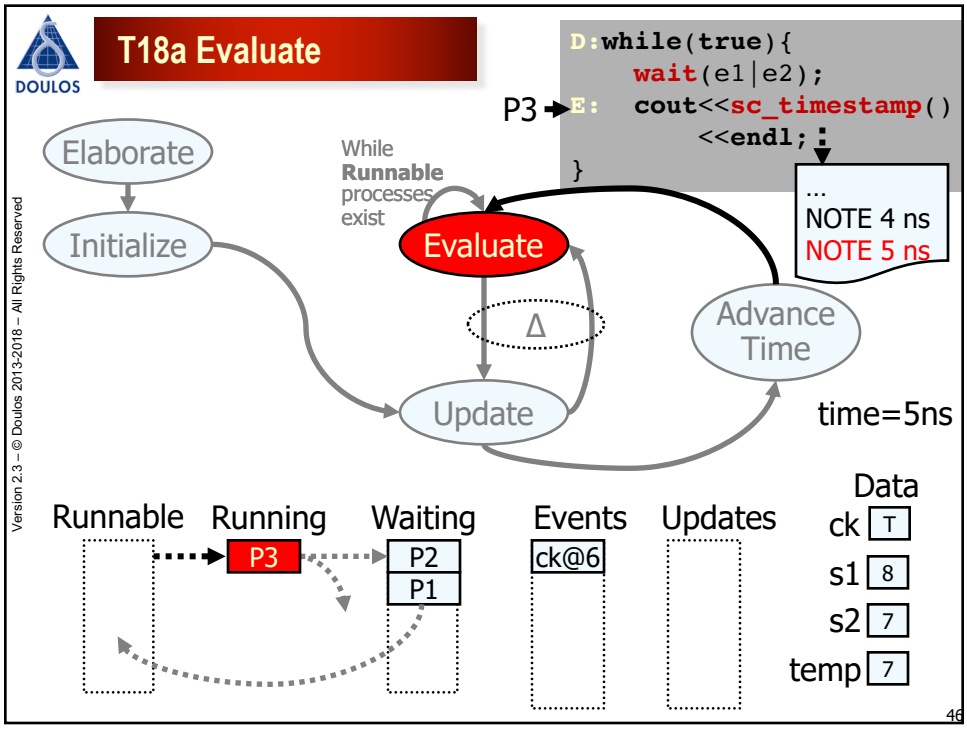

DOULOS
T14c Evaluate
P2
C:e1.notify(SC_ZERO_TIME);
wait();//static sensitive
i 7
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=4ns
Runnable
Running
Waiting
Events
Updates
Data
P3
P1
P2
e1@0
e2@5
ck@6
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
41

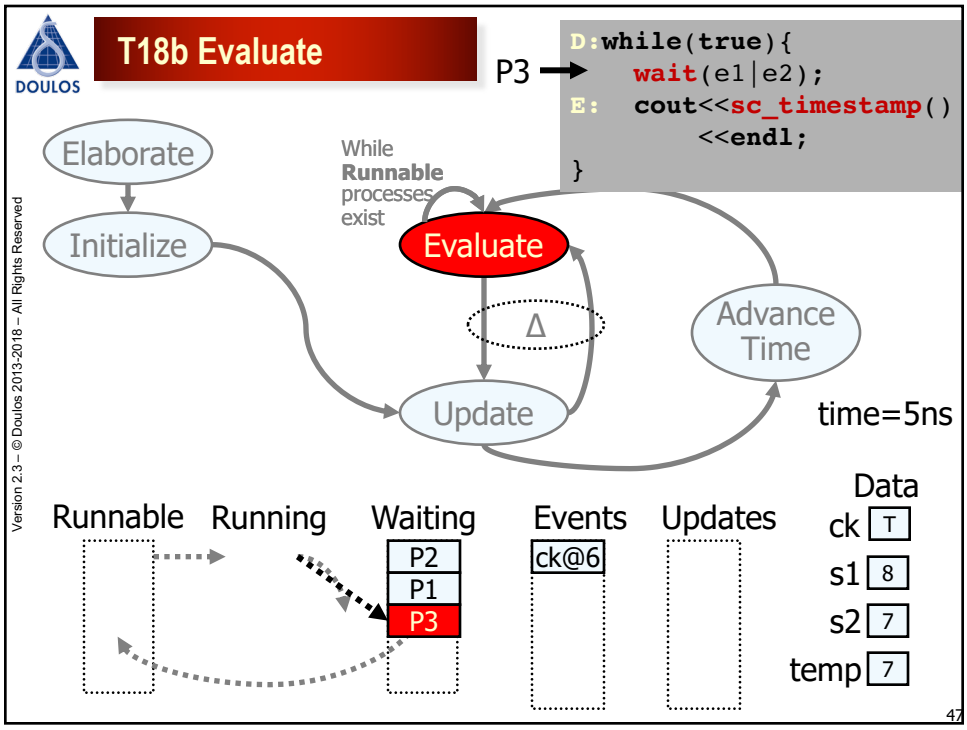

DOULOS
T15 Update
i 7
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=4ns
Runnable
Running
Waiting
Events
Updates
Data
P3
P1
P2
e1!
e2@5
ck@6
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
42

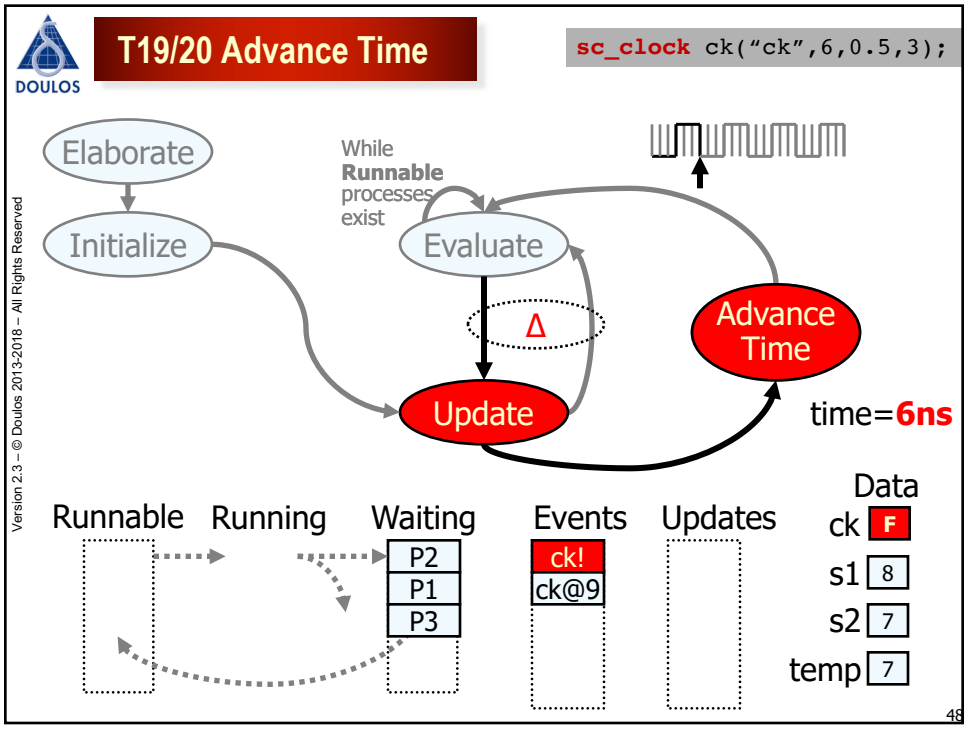

DOULOS
T16a Evaluate
D:while(true){
wait(e1|e2);
P3
E: cout<<sc_timestamp()
<<endl;
}
...
NOTE 2 ns
NOTE 4 ns
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=4ns
Runnable-To-Run
Running
Waiting
Events
Updates
Data
P3
P2
P1
e2@5
ck@6
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
43

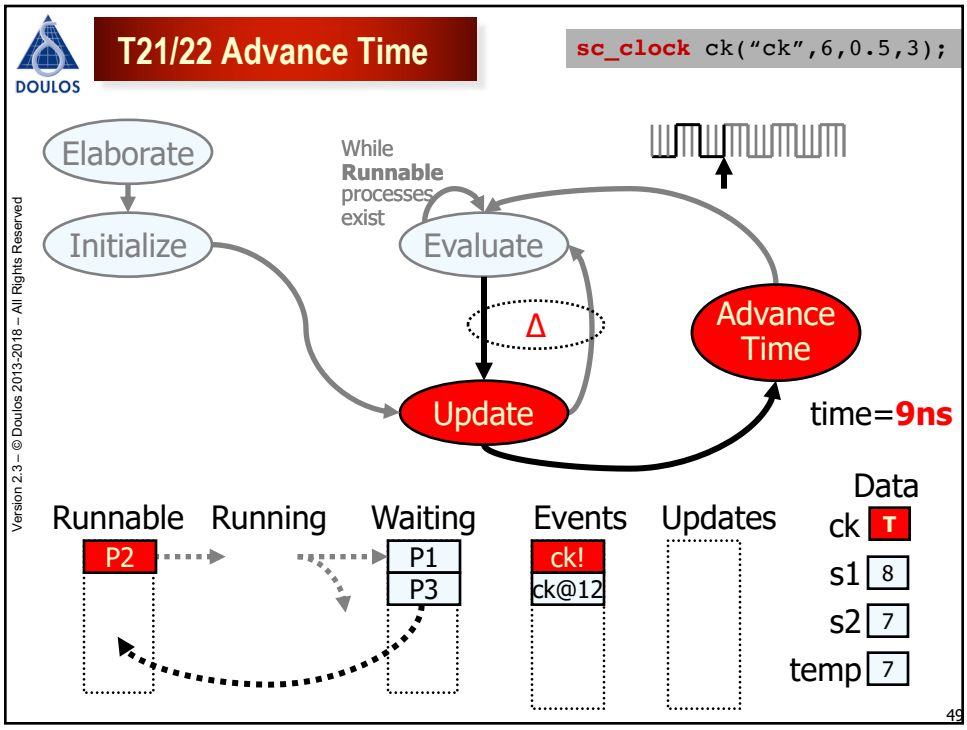

DOULOS
T16b Evaluate
D:while(true){
wait(e1|e2);
P3
E: cout<<sc_timestamp()
<<endl;
}
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=5ns
Runnable-To-Run
Running
Waiting
Events
Updates
Data
P2
P1
P3
e2@5
ck@6
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
44

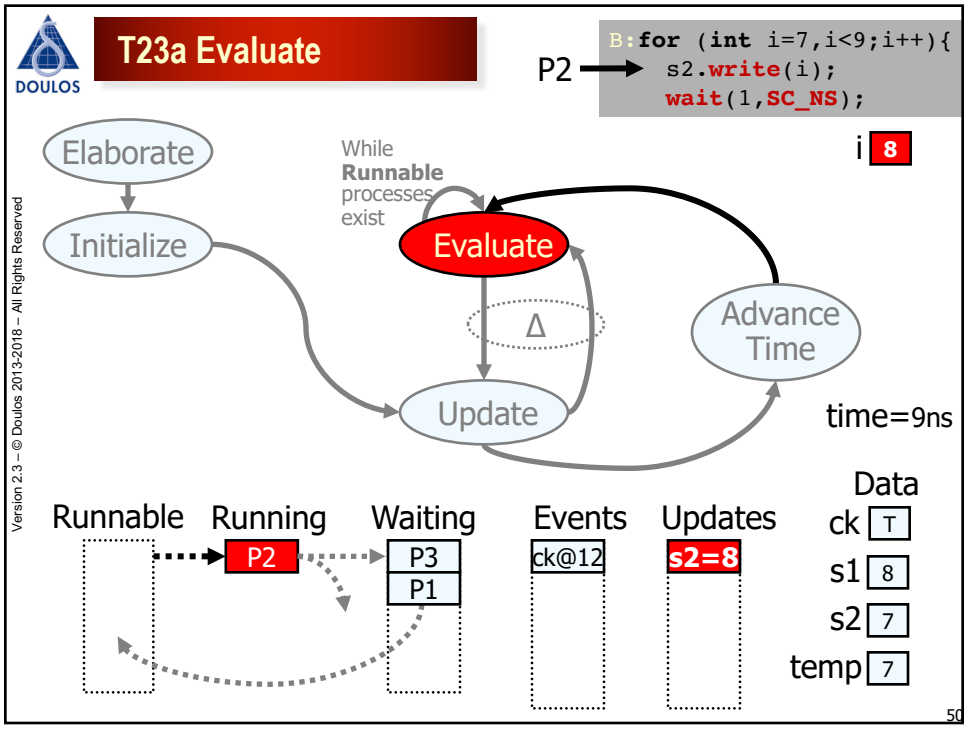

DOULOS
T17 Advance Time
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=5ns
Runnable
Running
Waiting
Events
Updates
Data
P3
P2
P1
e2!
ck@6
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
45


DOULOS
T18a Evaluate
D:while(true){
wait(e1|e2);
E: cout<<sc_timestamp()
<<endl;
}
P3
...
NOTE 4 ns
NOTE 5 ns
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=5ns
Runnable
Running
Waiting
Events
Updates
Data
P3
P2
P1
ck@6
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
46


DOULOS
T18b Evaluate
P3
D:while(true){
wait(e1|e2);
E: cout<<sc_timestamp()
<<endl;
}
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=5ns
Runnable
Running
Waiting
Events
Updates
Data
P2
P1
P3
ck@6
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
47


DOULOS
T19/20 Advance Time
sc_clock ck("ck",6,0.5,3);
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=6ns
Runnable
Running
Waiting
Events
Updates
Data
P2
P1
P3
ck!
ck@9
ck F
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
48


DOULOS
T21/22 Advance Time
sc_clock ck("ck",6,0.5,3);
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=9ns
Runnable
Running
Waiting
Events
Updates
Data
P2
P1
P3
ck!
ck@12
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
49


DOULOS
T23a Evaluate
B:for (int i=7,i<9;i++){
s2.write(i);
wait(1,SC_NS);
P2
i 8
Elaborate
Initialize
While
Runnable
processes
exist
Evaluate
Δ
Update
Advance
Time
time=9ns
Runnable
Running
Waiting
Events
Updates
Data
P2
P3
P1
ck@12
s2=8
ck T
s1 8
s2 7
temp 7
Version 2.3 – © Doulos 2013-2018 – All Rights Reserved
50

…and on and on it goes…

- Simulation returns when
  - No more **Runnable** nor **Waiting** processes
  - Encounters `sc_stop()` or `SC_REPORT_FATAL(…)`
  - Simulated `sc_pause()` or time exceeds specified `sc_start(tMAX)` time
  - Simulated time exceeds 64 bits
    - 2^{64} ps = 30 weeks 3 days 12 hours 5 min 44 sec
  - Really bad stuff
    - Explicit `exit()` or `abort()`
    - unhandled exception (control-C) or `kill`
    - bad memory access (pointer gone wrong)
    - out of memory (leak)
    - stack overflow

Observations

- Sample code does not follow best practices
  - designed to be short enough to fit on one page
  - only a single module
  - sc_in, sc_out discouraged
    - prefer sc_port< sc_signal_inout_if< T > >
    - unless you need the event finder specialization
  - sc_signal is a low-level construct (RTL)
  - sc_clock slows simulation - too much context switching
- Serious ESL simulations
  - involve 10's to 1,000's of processes
  - contain lots of module hierarchy and interconnect
  - use higher level of abstraction (TLM)
  - don't have explicit clocks

DOULOS

Process order of execution

- Processes model simulated concurrency (parallel execution)
- Each SystemC implementation has own initial ordering
  - important to allow reproducible results
  - should not depend on this behavior
- Can change this ordering by shuffling process registration
- Design required dependencies should be dictated by explicit events and handshakes

```
SC_CTOR(M)//force random
 …
 if (randomize) {
  switch (random()%3) {
   case 0: SC_THREAD(P3_thrd);
      break;
   case 1: SC_THREAD(P2_thrd);
      sensitive<<ckp.pos();
      break;
   case 2:
    SC_METHOD(P1_meth);
      sensitive<<s2;
      dont_initialize();
  }//endcase
 }//endif
}//end SC_CTOR
```

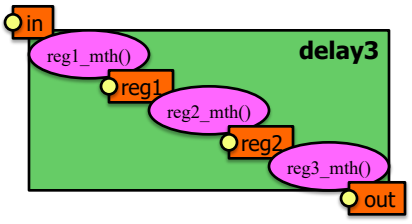

DOULOS

Guaranteeing order

- Would using simple int instead of `sc_signal<int>` work?

```
SC_MODULE(delay3) {
 sc_in<int> in_port;
 sc_out<int> out_port;
 sc_signal<int> reg1, reg2;
 void reg1_mth(void) {
   v1.write(in_port->read());
 }
 void reg2_mth(void) {
   v2.write(v1.read());
 }
 void reg3_mth(void) {
   out_port->write(v2.read());
 }
```

```
SC_CTOR(delay3) {
    SC_METHOD(reg2_mth);
    sensitive << in_port;
    SC_METHOD(reg1_mth);
    sensitive << in_port;
    SC_METHOD(reg3_mth);
    sensitive << in_port;
  }
};
```



How sc_signal works

- During update kernel calls update() method on all objects that did request_update() in preceding delta cycle
- write() is equivalent to verilog non-blocking assignment

```
mysig.write(expr);
myvar <= expr;
```

```
struct buffer
: sc_prim_channel, buffer_if {

 int read(void){return curr; }

 void write(int v) {
   next = v; request_update();
 }
 void update(void) {
   curr = next; evt.notify();
 }
 sc_event& written_evt(void) {
   return evt;
 }
private:
  int curr, next;
  sc_event evt;
};
```


For further study

- Download
  - Examine the basic code discussed
  - Examine the output log file
  - Examine the instrumented code used for the above
  - Includes the source to reproduce results
  - https://github.com/dcblack/engine
- Includes this presentation
- Try to get different results
  - Reorder the process registration code
  - Try a different platform, simulator vendor, or version

Common C++ Coding Pitfalls

- Forgetting semi-colon (;) on class/struct
- Forgetting to tag **`private`**, **`protected`** or **`public`**
- Copy-paste error on header guard (or completely forgetting)
  - HINT: Learn how to use #pragma once
- Failure to implement code separate from declaration
- Attempting construction inside the declaration
- Forgetting class-name qualifier when implementing separately
- Direct instantiation of class members leading to excess header needs
- Failure to use pass-by-reference (esp. for polymorphism)
- Abuse of pointers leading to memory faults
- Omitting the constructor or destructor
- Omitting copy-constructor or **`operator=`**
- Using **`printf`** instead of **`boost::format`** or **`fmt`**
- Failure to utilize STL or boost
- Not using const, and/or using #define

Common SystemC Coding Pitfalls

- Templating **`sc_port`** on non-**`sc_interface`** type
- Forgetting to bind (connect) all ports/sockets
- Forgetting to register processes inside constructor
- Ports use dot (**.**) operator instead of arrow (**->**) operator
- Using blocking functions inside SC_METHOD processes
- Incorrectly locating channels relative to **`sc_port`** or **`sc_export`**
- Coding at RTL level - too much detail
- Infinite loop path missing **`wait`** in SC_THREAD process
- Attempting to **`sc_stop`** and restart with **`sc_start`**
- Simulation phase actions during elaboration phase
- Using **`std::cout`** instead of **`SC_REPORT_INFO`**
- Using **`std::cerr`** instead of **`SC_REPORT_ERROR`**
- Elaboration phase actions during simulation phase
- Excessive context switching or I/O causing simulation to crawl
- Converting **`SC_THREAD`**s to **`SC_METHOD`**s to gain performance without first profiling code to determine real cause of slowdowns

SystemC Guidelines

- Abstract as high as possible – only add required details
  - Ask: What am I trying to answer with this model?
- Code as simply and cleanly as possible
  - State machines and wires are messy and hard to debug
- Avoid too much context switching (think)
- Be careful with immediate notification (delayed is safer)
- Communicate between processes with channels
- Communicate across module boundaries with ports
- Use indirect instantiation for flexibility (std::unique_ptr<>)
- Limit details - Model to requirements
- Improve your C++ coding skills

Questions

www.doulos.com