Chan and Patel

An annoted example of simulation

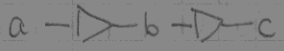
This is an in-depth walkthrough of example 1.6.5.2 in the notes

```
proc-a: process begin
   a <= '0';
   wait for 1ns;
   a <= '1';
   wait;
end process;
```

```
proc-b: process (a)
 begin
   b <= a;
 end process;
```

```
proc-c: process (b)
 begin
   c <= a;
 end process;
```

This is just the two-stage buffer we've seen before.



Initialization

Every process is set to resumed. Each signal is set to its default.

Time	0-
proc_a	R
proc_b	R
proc_c	R
a_{proj}	U
a_{actual}	U
b_{proj}	U
b_{actual}	U
c_{proj}	U
c_{actual}	U

This is pretty simple so far. This initialization happens before time actually starts to tick, which is why I've labelled time as 0-.

From here on in, I'll just be drawing the parts of the table that are relevant.

Simulation Begins!

We now increase our time to 0ns. Proc_a gets executed first (though remember, the order doesn't really matter). One column = one action taken (mode change / signal change)

Time	0-	0		
proc_a	R	E		S
a proj	U		0	
a actual	U			

Now, we have proc_b: since it's resumed, it must be executed, copying a actual's 'U' value into itself. Same with proc_c.

Time	0-	0						
proc_b	R		E		S			
proc_c	R					E		S
bproj	U			U				
bactual	U							
cproj	U						U	
cactual	U							

Now, all processes are suspended, which means we can go to the next delta cycle. First, the only changed value is a, so we copy over the projected to a actual. This then prompts a change in proc-b, which relies on a's value

Proc_b sees the change, gets set to resume, begins executing to calculate bproj, then suspends

↙ extra stuff here

Time	0 ···	0				
proc_b	S		R	E		S
aactual	U	0				
bproj	U				0	
bactual	U					

↖ start of delta cycle

At this point, all processes are suspended, so we begin another delta cycle, copying bproj to b actual. This prompts proc_c to resume, and calculate Cproj

Time	0 ···	0				
proc_c	S		R	E		S
bactual	U	0				
Cproj	U				0	
Cactual	U					

Again, all of our processes are now suspended. We begin the next delta cycle, with Cproj getting copied.

Time	0 ···	
Cproj	0	
Cactual	U	0

Now, all processes are suspended, AND there are no more projected values to copy, so we increase the time. Proc-a is done waiting, and it resumes, changing aproj to 1.

Time	0	1			
proc-a	S	R	E		S
aproj	0			1	
a actual	0				

And now, this whole process is repeated again:

1) All suspended, so aactual <= aproj
2) Proc-b resumes, calculating bproj
3) All suspended, so bactual <= bproj
4) Proc-c resumes, calculating cproj
5) All suspended, so cactual <= cproj
6) No more projected values to copy, increment time to 2ns
7) All processes suspended, no more changes in values

Let's take a look at the simulation

			(1)	(2)	(3)	(4)	(5)	(6)	(7)
Time	0	1...	1					2	
proc-a		S						S	
proc-b		S		R E S				S	
proc-c		S				R E S		S	
aproj		1						1	
aactual		0	1					1	
bproj		0		1				1	
bactual		0			1			1	
cproj		0				1		1	
cactual		0					1	1	

And that's it! Hopefully this explanation has been useful.