

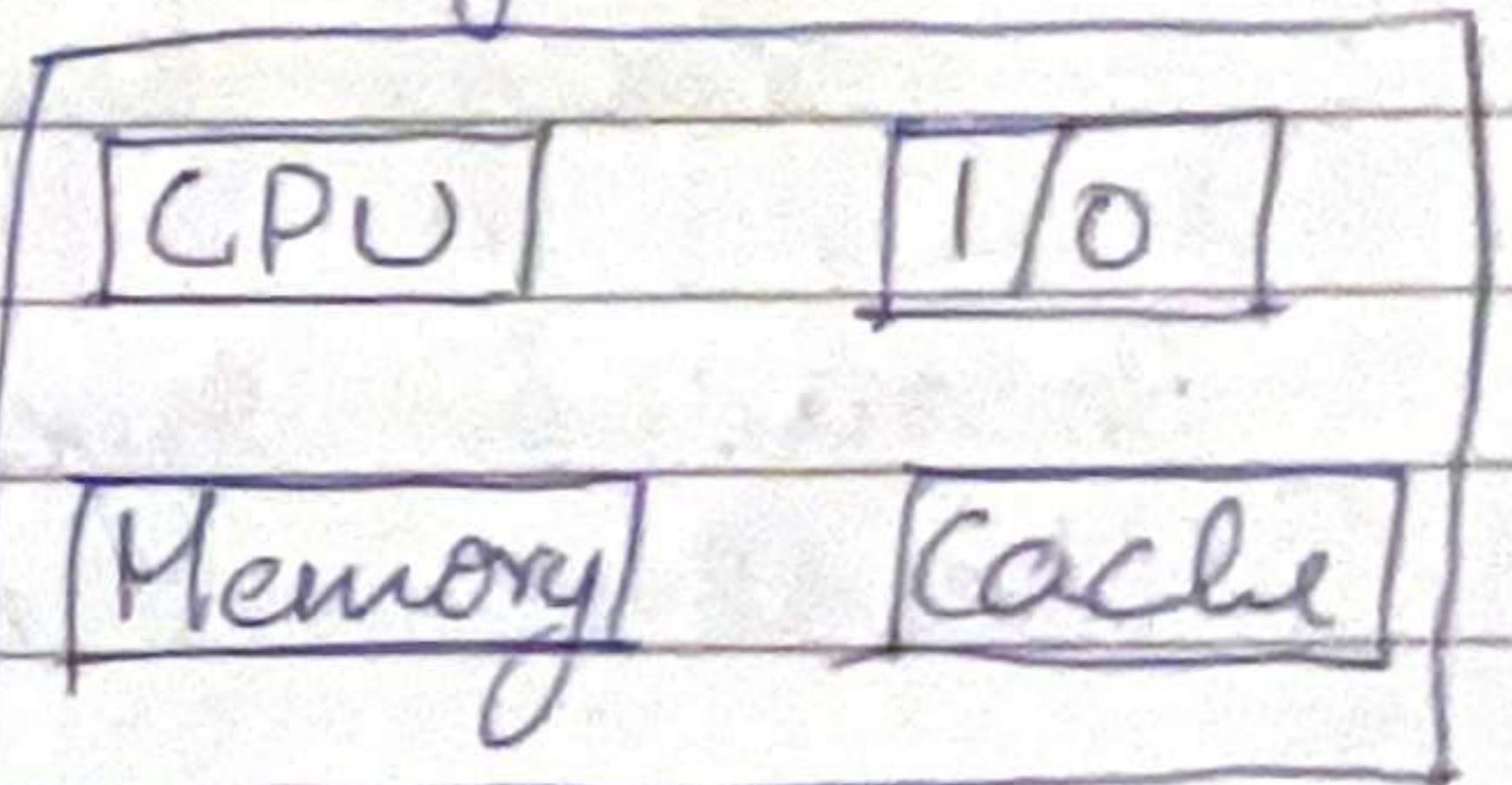
Embedded System Modeling

→ Lecture 1

Computers are everywhere

System - on - chip (SoC)

- Design of a complex embedded systems on a single chip
- reduced cost
- easier to build the system



Systems embedded into cyber-physical systems

- constraints for external ip (real time)
- Application specific (not general purpose)

challenges to design:

- 1) often mobile (Battery powered)
- 2) highly reliable (can stand ↑ temp)
- 3) high performance constraints
(are often real-time)

4) high complexity.

Advantages

- application known at design time
since it is app specific.
- environment known at design time.
- customization.

SOC design faces tremendously high design complexity.

Abstraction layer.

Level no of components

System	$1e^0$	↑ Abstraction	↓ Accuracy
Algorithm	$1e^{1-2}$		
RTL	$1e^{3-4}$		
Gate	$1e^{5-6}$		
Transistor	$1e^{6-7}$		

Move to higher level of abstraction.

Product features

Specification model



Architecture model



Communication model



Implementation model

} top-down system design flow.

Model

It is an abstraction of reality.

- some features explicitly present
- simplified features
- some features are omitted

Modeling

It is the creation of model.

The refinement of model.

Model of computation

It has an underlying mathematical model behind it.

1) Programming models.

- C/C++, Java, Python

2) Process based

- Processes and threads
- Kahn Process model
- Synchronous data flow

3) State based models.

- FSM, DFG, FSMD, SFSMD, HCFSM

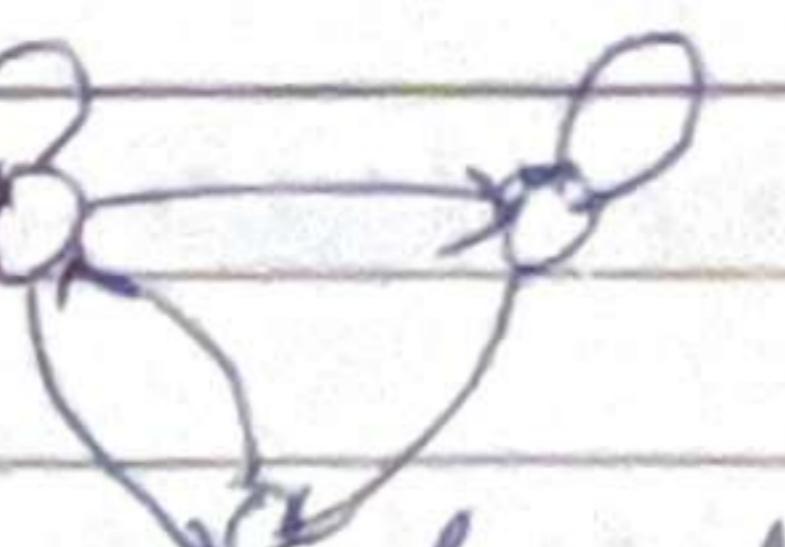
Computational model

* Formal, abstract description of a computing system

with multiple

- supported features
- complexity
- expressive power

- FSM



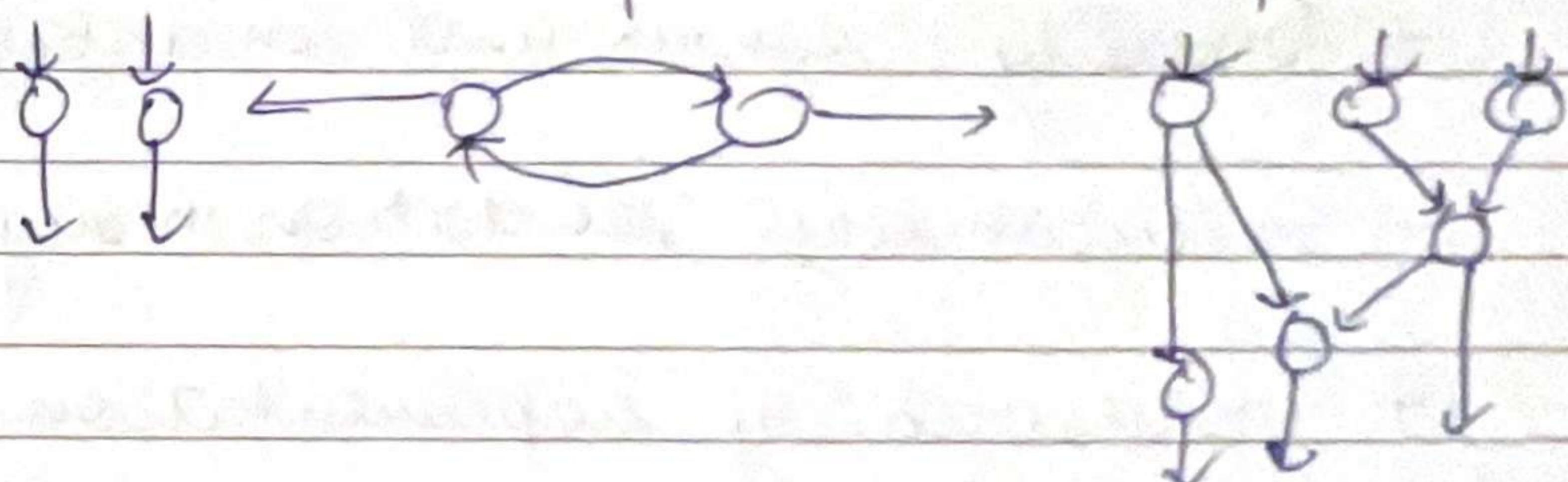
Basic model for describing control

- DFG Data flow Graph

Basic model for describing computation & data flow.

- FSMD finite state machine with data

Combined model for control & computation



- SFSMD Super-state FSM with data

States are described by procedures in programming language

- HCFSM Hierarchical concurrent FSM

multiple FSM composed hierarchically and in parallel

• Program State Machine (PSM)

states described by procedure in a programming language.

eg : Spec C.

System level Description Language

- Goals and requirements.
 - formality : syntax and semantics.
 - Executability: validation through simulation
 - Synthesizability: implementation of HW & SW.
 - Modularity: separation of concepts.
 - completeness: support for all concepts found in embedded system.
 - Orthogonality:
 - Simplicity

SLDL - orthogonal Concepts & constructs

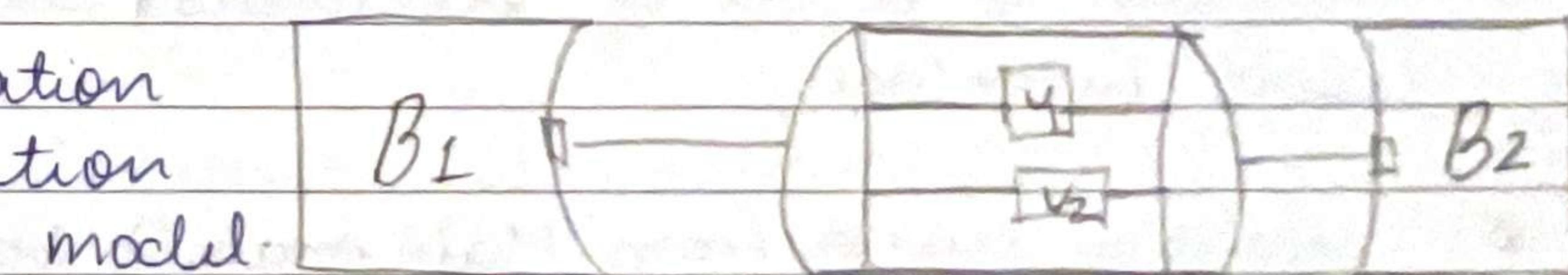
(SLM) System level Modeling

- computation
encapsulated in behaviors
- communication
encapsulated in channels

Spec C.

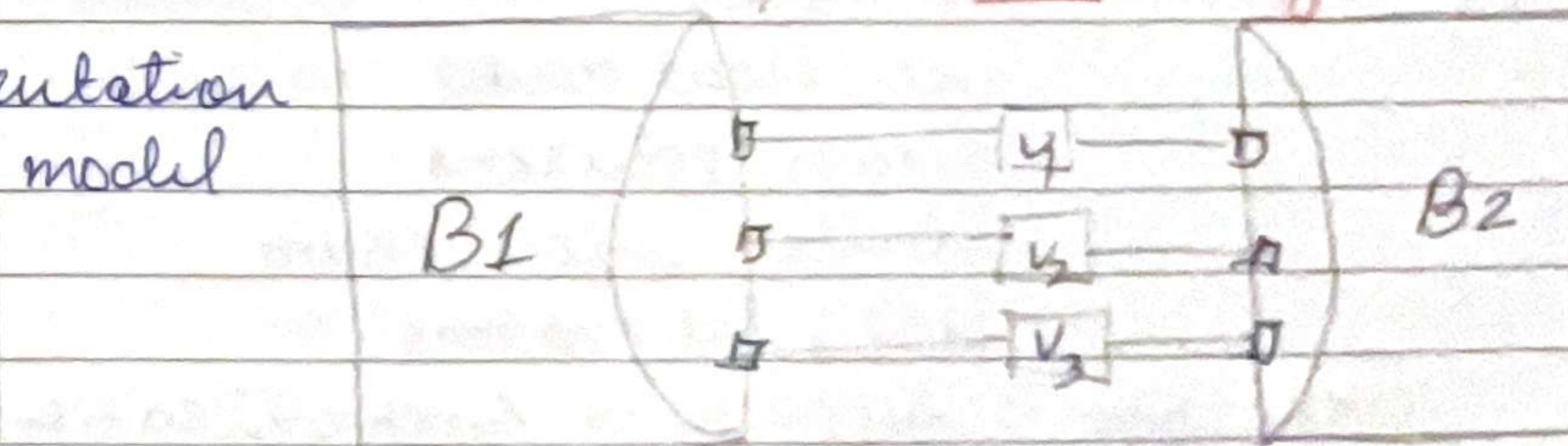
- * Behavior and channels.
- * Separation of computation & comm
- * plug and play

- specification
- exploration
- model



protocol inlining

- implementation
- model



channel disappears
communication inlined into behavior
wires are exposed

→ Lecture 3.

Spec C language

* foundation : ANSI - C

• Spec C is a true superset of ANSI - C

• every C program is a Spec C program

• Spec C has extensions needed for hardware

• Spec C is a real language.
relies on dedicated compiler & static analysis

* Program is a set of behaviors, channels
and interface

* execution starts from Main.main() behavior.

* supports all types.

int, float, double

arrays, pointers

struct, union, enum

bool, bit vectors

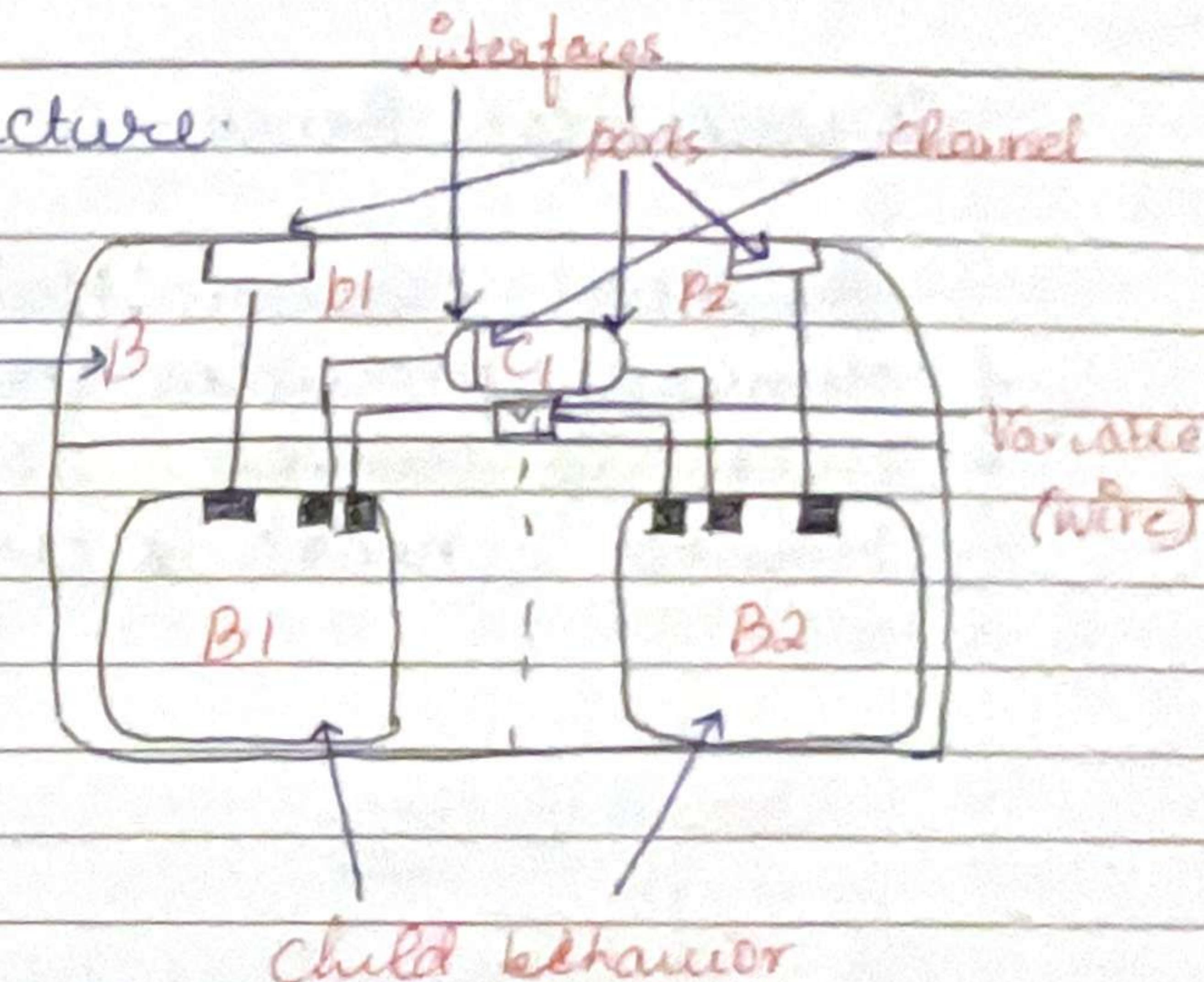
event type : support of synchronisation.

event e

Buffered & signal type : explicit support of
RTL concepts.

Basic structure:

Behavior



Structural hierarchy:
interface I1

{,

channel C1 implements I1

{,

Behavior B1 (

behavior B (

{,

void main (

{ par { b1;

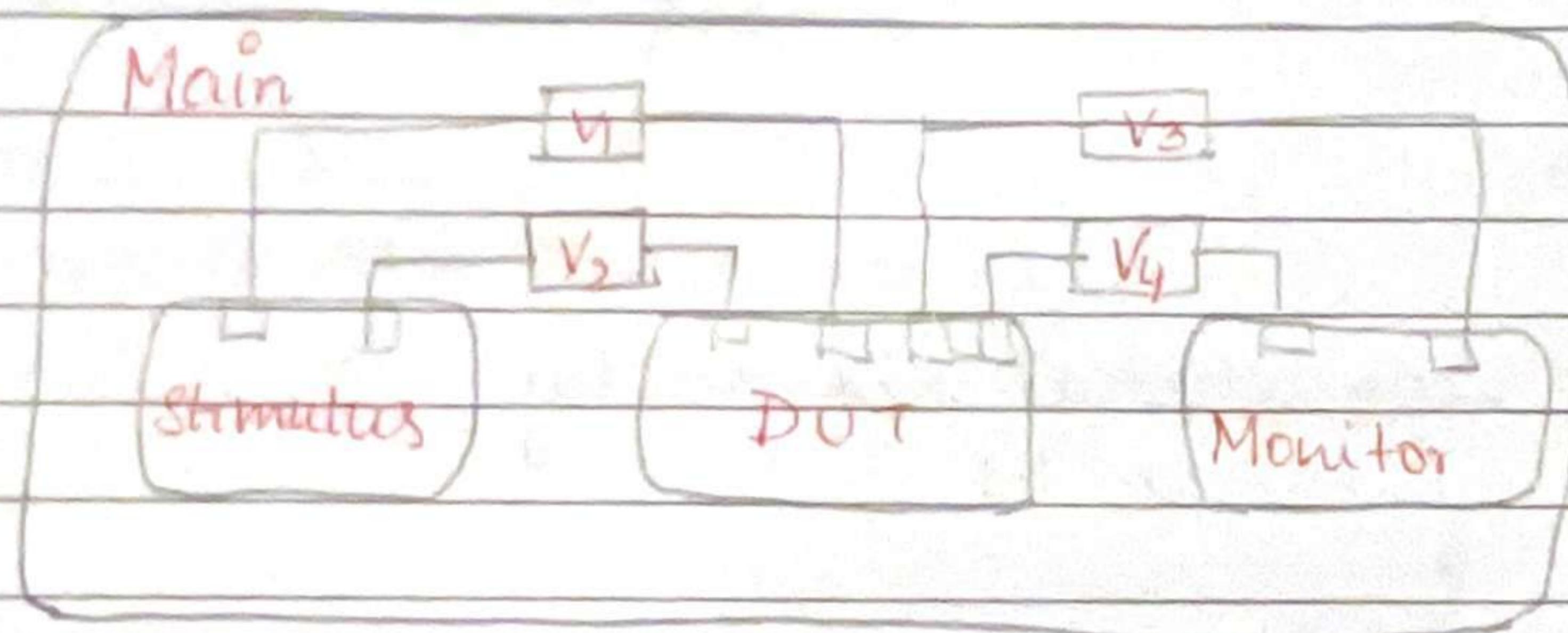
b2;

{,

{;

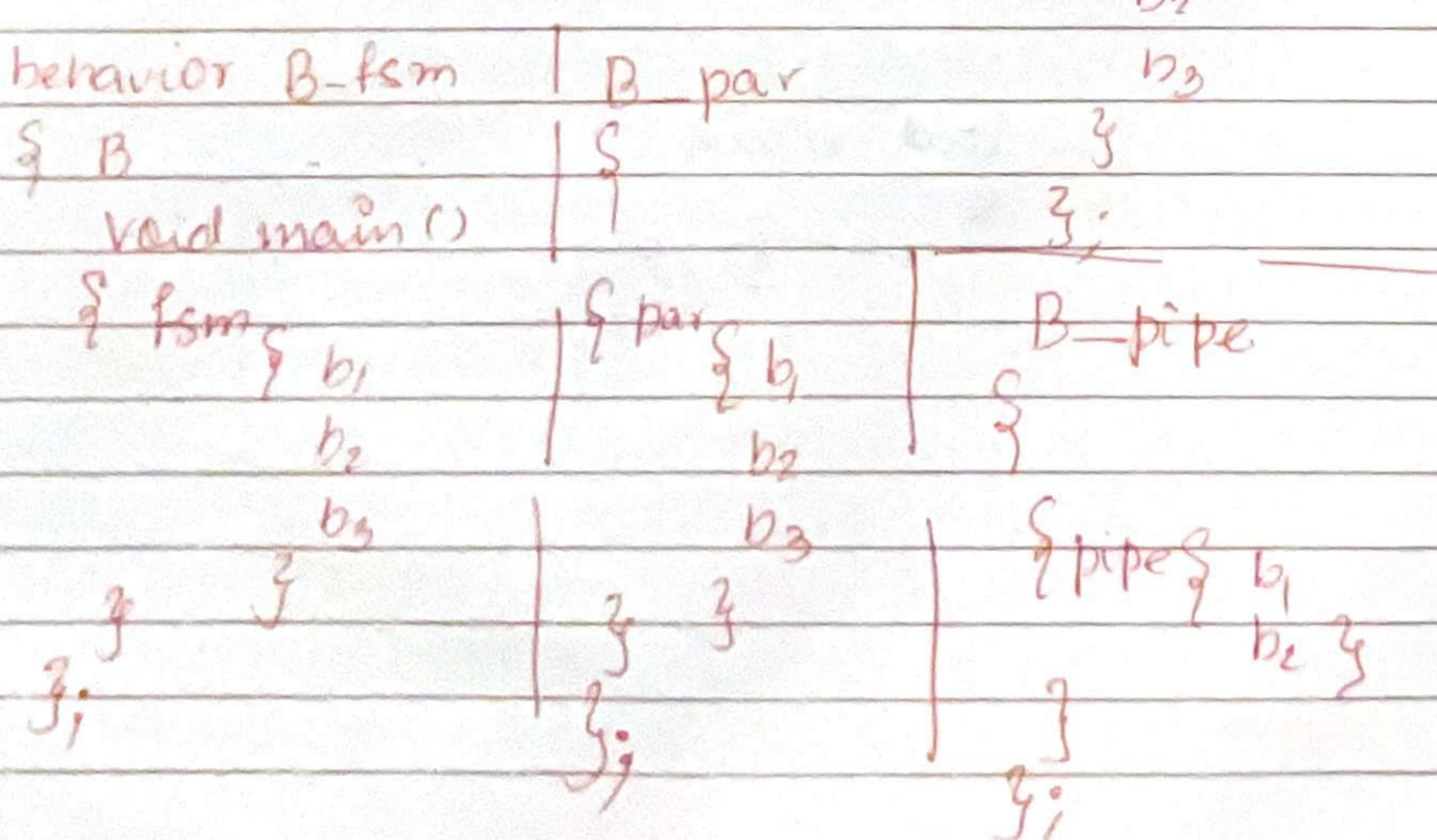
Typical test Bench

- Top level behavior : Main stimulus provides test vectors
 - DUT is target Soc
 - Monitor checks & observes output



Behavioral hierarchy

- * Sequential execution
 - * FSM execution
 - * Concurrent execution
 - * pipelined execution.



Pipeline

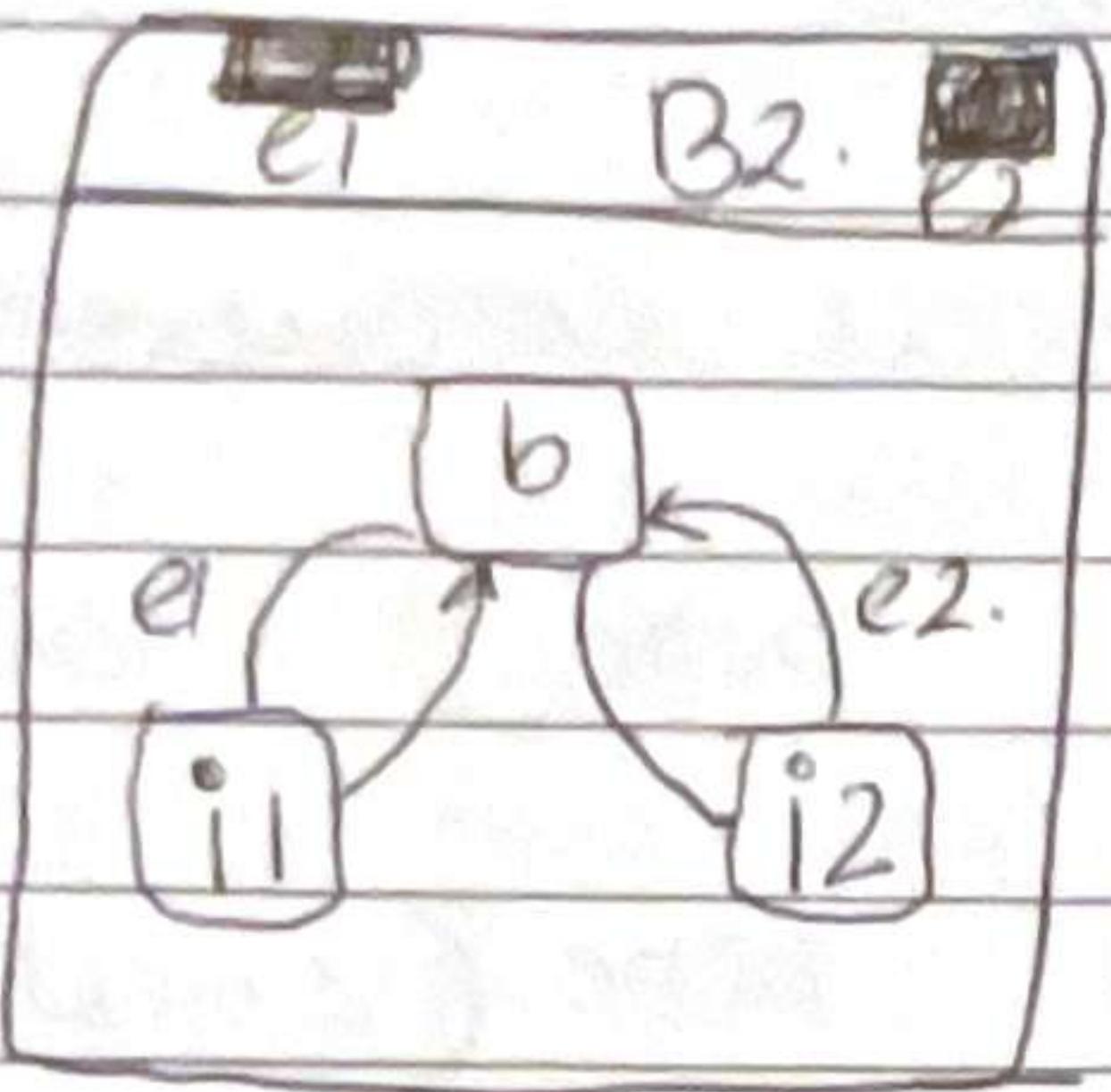
- + Explicit execution in pipeline fashion
pipe { <instance-list> } ;
pipe { <init>; <cond>, <incr> } { } ;
 - + Supports automatic buffering
piped [...] <type> <variable-list> ;

Exception handling.

- Abortion

behavior B1 (en event e1
in event)
{ B b, a1, a2
void main (void)
{ try { b; }
trap (e1) { a1; }
trap (e2) { a2; }
}
}
}

- interrupt



Void main (Void)

{ try { b; }

interrupt (e1) { i1; }

interrupt (e2) { i2; }

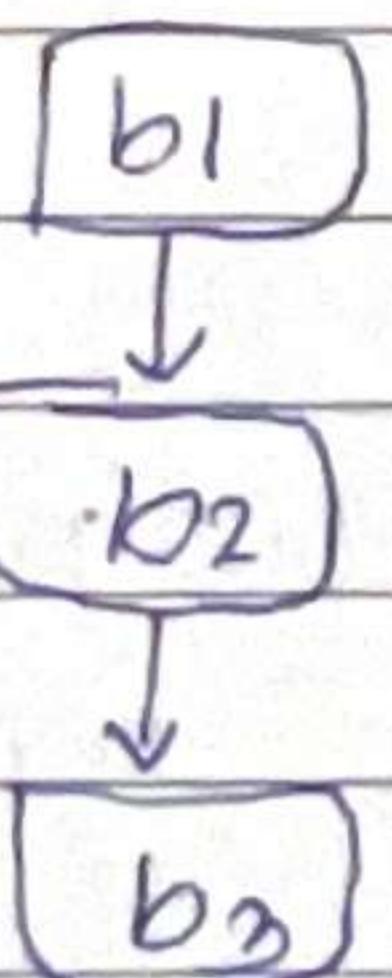
}

→ Lecture 4

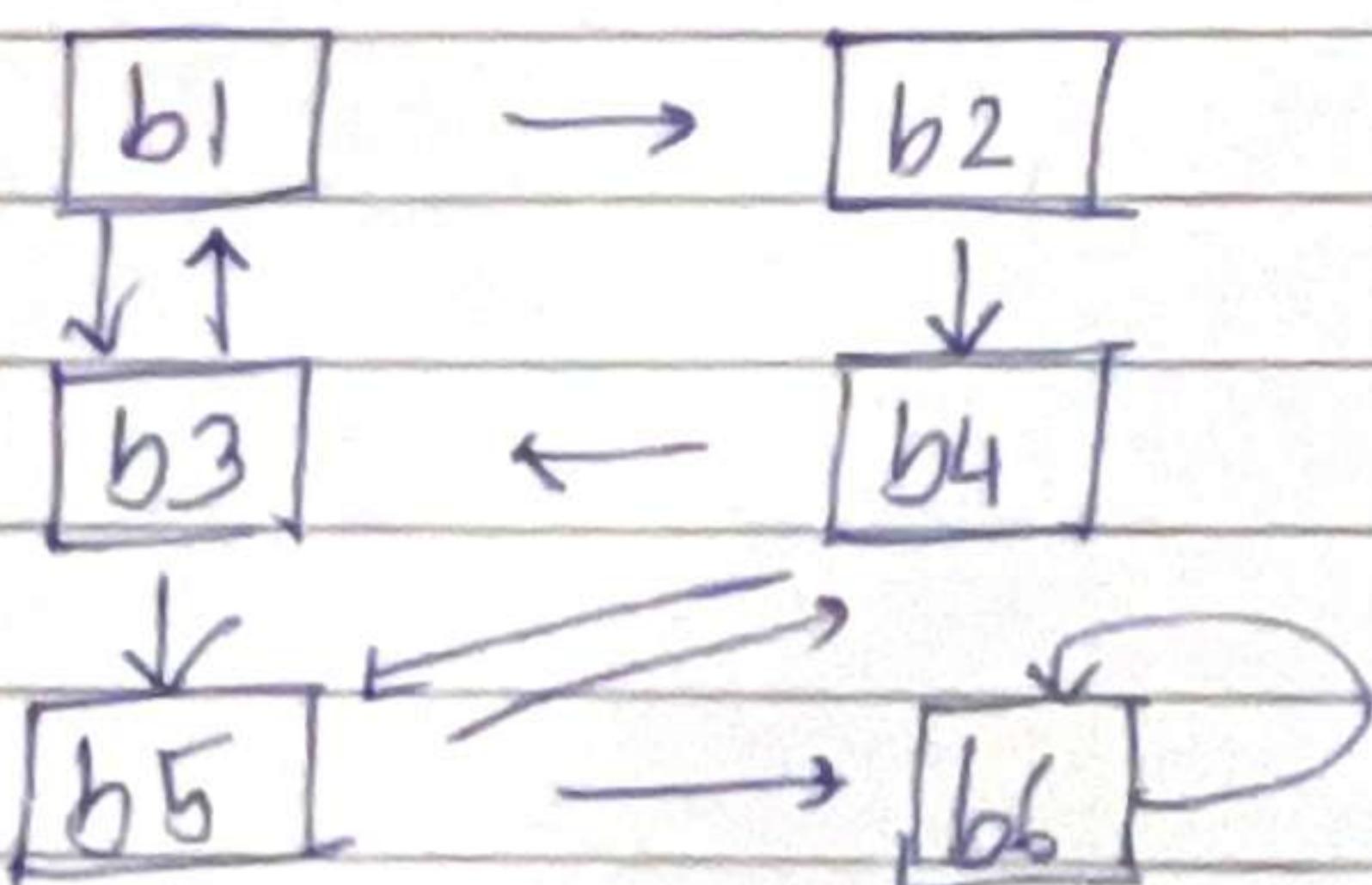
~~Spec.~~
System C

• Behavioral hierarchy

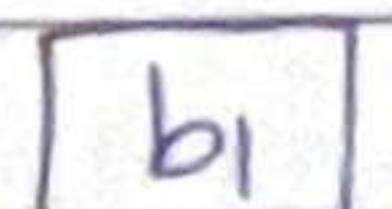
1) Sequential execution



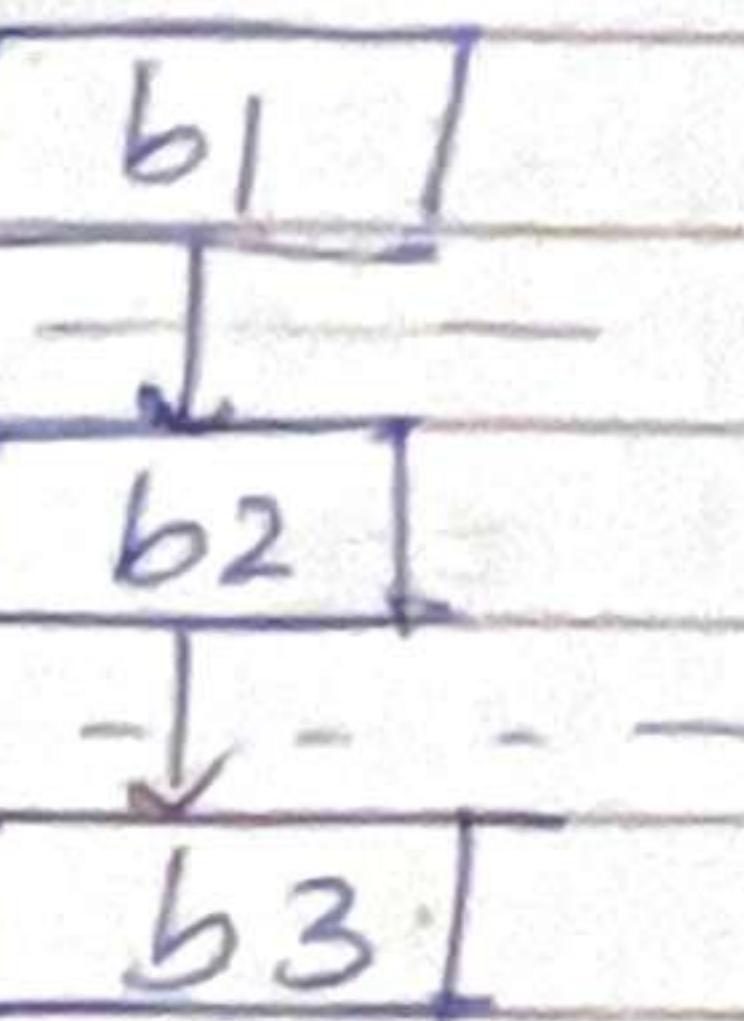
2) FSM.



3) concurrent



4) pipeline



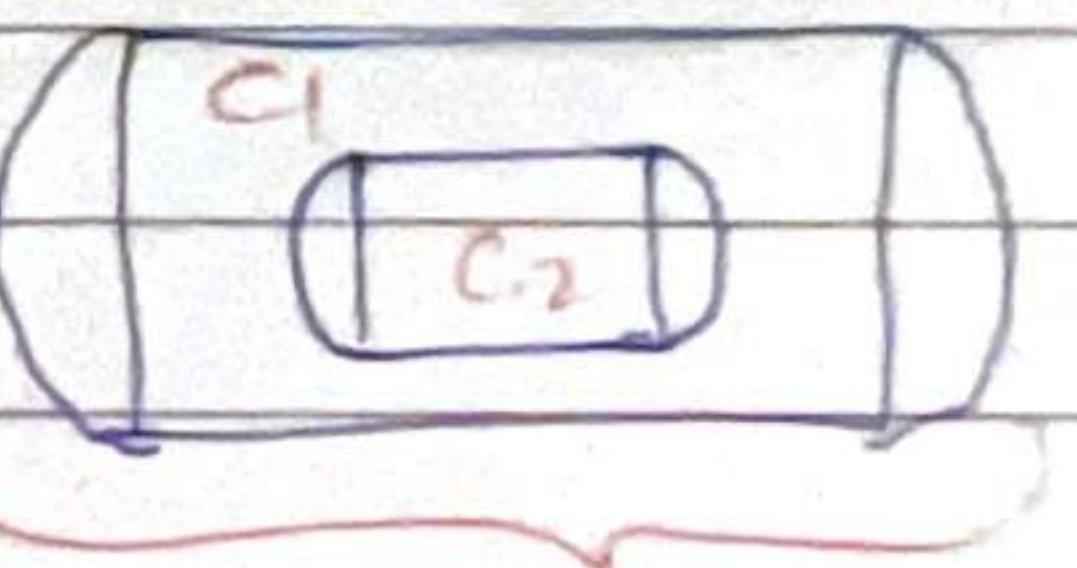
• Communication and synchronisation

Message passing

- Via Shared Variable
- via channel & interfaces

can also happen

- via hierarchical channel.



protocol stack

Synchronisation happens → using

- wait
- notify
- ACK

Behavior / Event wait for a sequence and then wait for 'ACK' once it is sent by the other event

- * how to define a channel or interface.

⇒ interface <name>
 { <declaration> }.

⇒ channel class

channel <name>

implements < interfaces >
 { < implementation > }.

- * When using shared memory, you need Semaphores.

→ Semaphore
 Synchronisation parameters. Semantics.
 Synchronisation parameters. Semantics.

- It is a unsigned integer.
- changes are atomic.
- can only have 2 operations.

Wait()

post().

wait()

It decrements the semaphore, unless it's value is 0

When it 0, it waits.

Post()

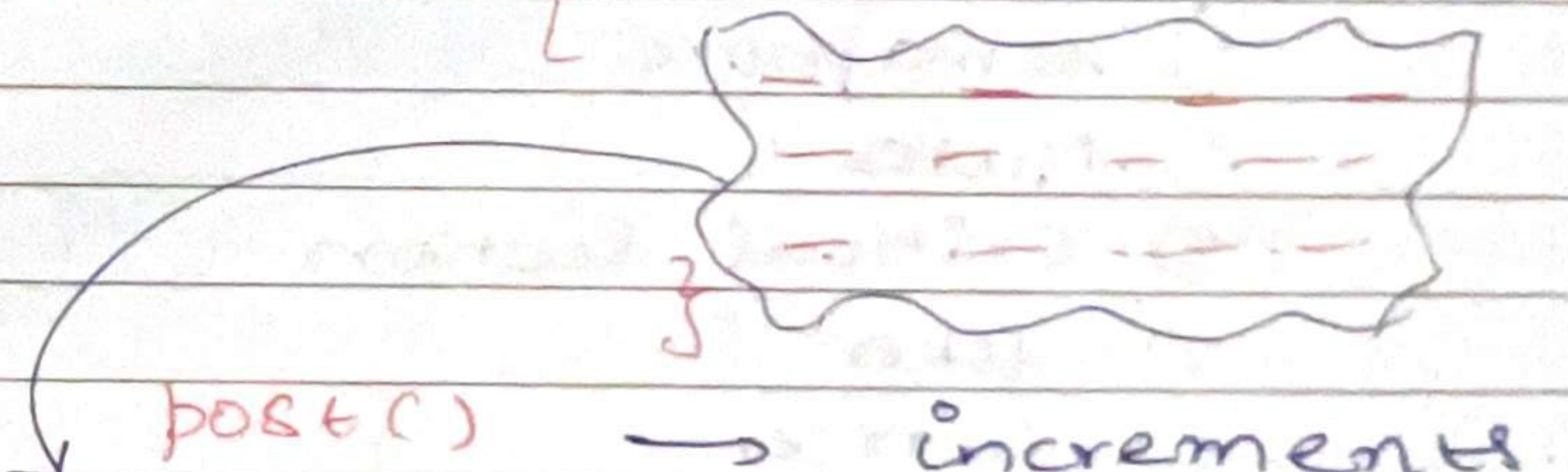
increments the value.

Binary Semaphore

↳ They act as locks.

→ Value 0 or 1

Wait() → decrements.



post() → increments.

Critical Section

→ Mutually exclusive executions.

- 1) Semaphores.
- 2) Mutex.

* Single handshake is for synchronisation.
* double handshake is for communication.

→ standard channels.

* Import synchronization channel.

- 1) semaphore
- 2) Mutex
- 3) Critical Section
- 4) token
- 5) barrier
- 6) Handshake

#

* type-less communication channel.

- 1) double handshake
- 2) Queue

* typed communication channel

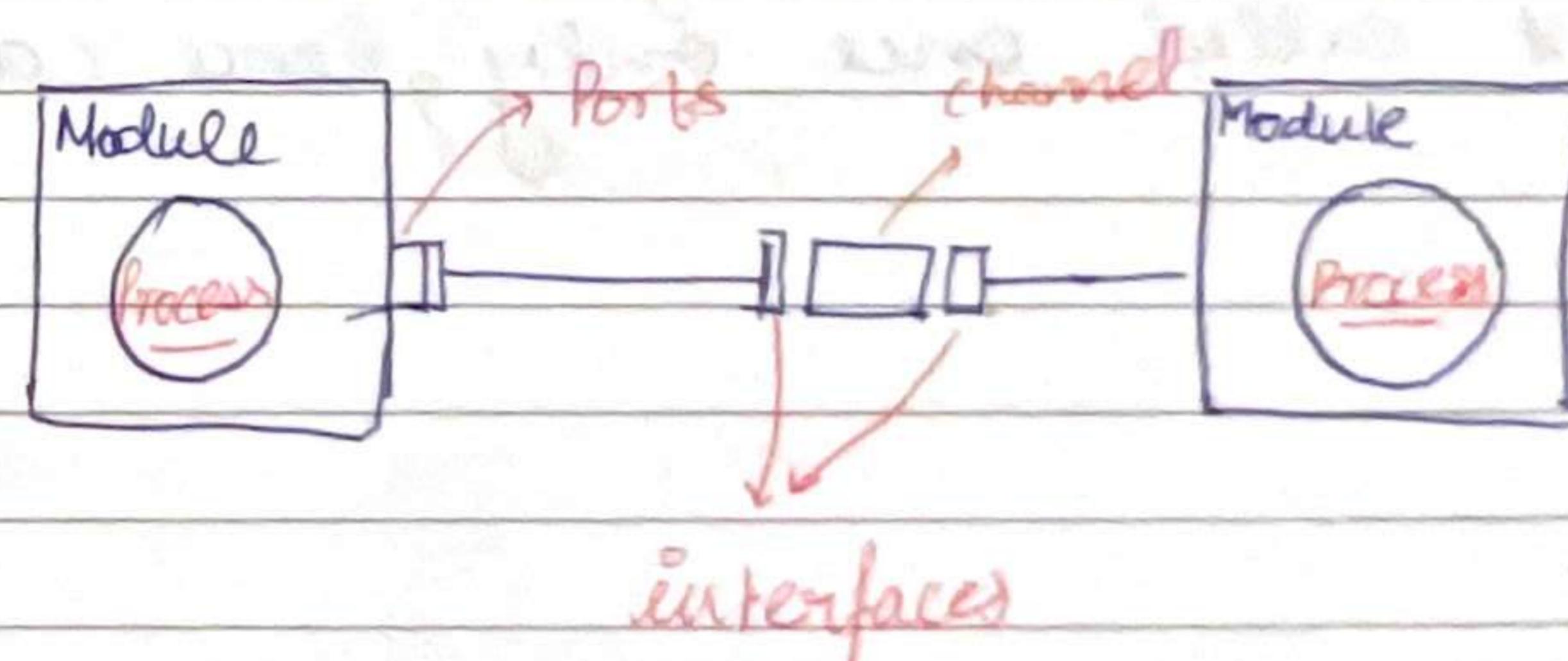
- 1) double handshake
- 2) queue

→ Lecture 6

System C 231
↳ recent

- * developed from C++
- * Abstraction levels

- untimed model
- Transaction level model
- Bus functional Model
- Cycle accurate model



ports are bidirectional since no direction is mentioned

SystemC R → library.

SC - Module } → class.

→ Process in SystemC

SC - method.

- * fast or O - time.
- * can not contain infinite loop.
- * can not be suspended.

SC - thread.

- * slower, but powerful.
- * can have infinite loop.
- * called once only, hence can have ∞ loop.

→ Lecture 9

Essential concepts in embedded system
Modeling

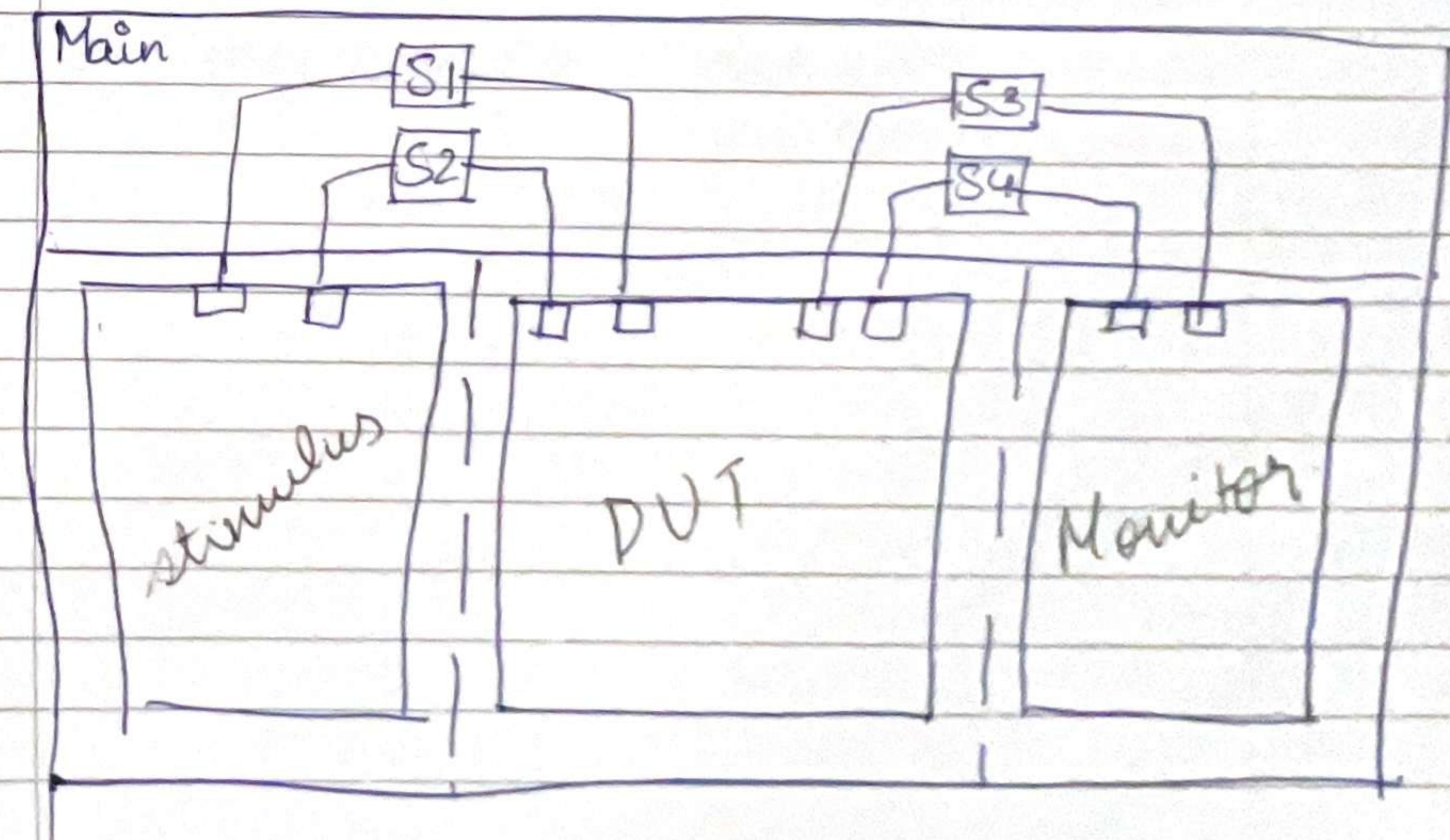
- Behavioral hierarchy
- Structural hierarchy & connectivity.
- Synchronisation and communication
- Timing.

→ Lecture 10

Test Bench Modeling.

Create

- Stimulus
- DUT
- Monitor.



* Bus functional Model

- * We simulate the test bench, but don't synthesis it.
- * We simulate & synthesis the DUT.

~~Lecture 11~~

→ Lecture 12

Computation

- # **Granularity** - leaf behavior
- # **hierarchy** - Explicit execution order
- # **encapsulation** - localised variable explicit port mapping
- # **concurrency** - Potential parallelism
- # **timed** - untimed

Communication

- # **Comm** - std channel lib.
- # **Synch** -
- # **Dependencies** - data flow explicit in connectivity

How to improve speeds

- pipelining
 - Parallelization
 - hardware optimisation
 - Software
 - app
- Lec 17

* Real time goal - No. how far off - 211x. $\gamma \underline{A7}$

$$1/3890 = 257 \text{ FPS}$$

~ ~ ~

Assignments

Assignment

A1

Goal: Design a suitable embedded system model of this application & describe it using SLDL.

1) Bug fix in NMS.

1 col and row is being omitted, after that

2) All warning must be cleared as we go so that they do not form problems in the future work.

Warning can become a bug later.

A2

1) Create prod cons behavior which are parallel

A3

1) prod - cons in SystemC.

Soc supports initialisation of variables with expression
that are constant at compile time.

→ A4 creating the SL specification model

1) The Macro NULL is not defined,
so change it to Zero '0'

2) Flexible configuration parameter
Should be converted to constants
time

constants malloc () {
 ↑ code reduce
 memory fragmentation
 calloc () } not feasible during
 free () hardware implementation
↓ ↓
Memory increase performance

* SoC can not be represented as
a new memory chip at run time

* Command line para. can only
be used with test bench, not
the SoC. eg: filename

3) Window size = 21

sigma, rows, cols

max sigma = 4 ← replace with 4

Windowsize = $1 + 2 * \text{ceil}(2.5 * \text{sigma})$

$$= 1 + 2 * \text{ceil}(10)$$

$$= 1 + 2 * 10$$

$$= 21$$

=

4) radian direction & angle radians

↳ radiant direction image

Since we only need edge image, we
do not require these 2 functions.

→ A5

1) Convert application to process video
instead of picture

2) pixel to (2704×1520) device.
↓
higher memory for our application

array variables holding our data

3) adjust stack overflow

Otherwise get segmentation fault

We adjusted stack space allocation
in linux.

Stack size → 128 ~~Mbytes~~ Mbytes.

4) Create test bench.

* Stimulus, platform, Monitor
in parallel
data in DUT data out.

Better integration & testing
platform can abstract away the complexities of handling
diff. dt input types

- * data in & data out, allows our test
bench to remain un-modified
even when communication to DUT
is implemented by a BUS

- * Our target SoC will never stop,
we even an endless loop.
unless power is turned off.

→ A6 start designing our target
implementation.

- 1) all function become behaviors.

* Sequential execution. { Reduce
pipeline idea } bottleneck
starts. { bottleneck
neck }

- 2) refine gaussian smooth { Bottleneck }

→ A7 Performance
estimation

- * calculate the time using $\langle \text{time} \cdot h \rangle$

get time in seconds & then
convert to %.

System C

→ because sysc is pipelined by default

→ A8

- 1) Track the delay between frames in ms

- 2) Add delay for R-pi, to replicate Real time Values.

Real time embedded Values

- 3) calculate the throughput

We can operate on multiple images

(FPS)

- 4) pipeline DUT

Bottle neck
start ends

- 5) Decompose GS

↳ Blur X
↳ Blur Y

After decomposing Blur Y is the
Bottleneck

- 6) Slice Blur X and Blur Y

parallelise the functions

Now, NMS becomes the RPi

→ A9

1) Optimisation - O.

- * We don't always write optimised code because sometimes compiler may remove important part of the code.

2) ~~#~~ $25 \times$ speedup after optimisation.

3) R-pi 3 to R-pi 4.

↪ Better hardware.

4) floating point to fixed point

→ faster execution speed → cheaper
NMS → Bottleneck. ↳ faster
acceptability loss
→ floating point was too slow. ↳ with
acceptable loss of accuracy

* To ↑ throughput by ↓ load on pipeline stages.

* need to optimise long stage delay.

11-15
12-15
13-15
14-15
15-16
16-17
17-18
18-19
19-20
20-21
21-22
22-23
23-24
24-25
25-26
26-27
27-28
28-29
29-30
30-31
31-32
32-33
33-34
34-35
35-36
36-37
37-38
38-39
39-40
40-41
41-42
42-43
43-44
44-45
45-46
46-47
47-48
48-49
49-50
50-51
51-52
52-53
53-54
54-55
55-56
56-57
57-58
58-59
59-60
60-61
61-62
62-63
63-64
64-65
65-66
66-67
67-68
68-69
69-70
70-71
71-72
72-73
73-74
74-75
75-76
76-77
77-78
78-79
79-80
80-81
81-82
82-83
83-84
84-85
85-86
86-87
87-88
88-89
89-90
90-91
91-92
92-93
93-94
94-95
95-96
96-97
97-98
98-99
99-100
100-101
101-102
102-103
103-104
104-105
105-106
106-107
107-108
108-109
109-110
110-111
111-112
112-113
113-114
114-115
115-116
116-117
117-118
118-119
119-120
120-121
121-122
122-123
123-124
124-125
125-126
126-127
127-128
128-129
129-130
130-131
131-132
132-133
133-134
134-135
135-136
136-137
137-138
138-139
139-140
140-141
141-142
142-143
143-144
144-145
145-146
146-147
147-148
148-149
149-150
150-151
151-152
152-153
153-154
154-155
155-156
156-157
157-158
158-159
159-160
160-161
161-162
162-163
163-164
164-165
165-166
166-167
167-168
168-169
169-170
170-171
171-172
172-173
173-174
174-175
175-176
176-177
177-178
178-179
179-180
180-181
181-182
182-183
183-184
184-185
185-186
186-187
187-188
188-189
189-190
190-191
191-192
192-193
193-194
194-195
195-196
196-197
197-198
198-199
199-200
200-201
201-202
202-203
203-204
204-205
205-206
206-207
207-208
208-209
209-210
210-211
211-212
212-213
213-214
214-215
215-216
216-217
217-218
218-219
219-220
220-221
221-222
222-223
223-224
224-225
225-226
226-227
227-228
228-229
229-230
230-231
231-232
232-233
233-234
234-235
235-236
236-237
237-238
238-239
239-240
240-241
241-242
242-243
243-244
244-245
245-246
246-247
247-248
248-249
249-250
250-251
251-252
252-253
253-254
254-255
255-256
256-257
257-258
258-259
259-260
260-261
261-262
262-263
263-264
264-265
265-266
266-267
267-268
268-269
269-270
270-271
271-272
272-273
273-274
274-275
275-276
276-277
277-278
278-279
279-280
280-281
281-282
282-283
283-284
284-285
285-286
286-287
287-288
288-289
289-290
290-291
291-292
292-293
293-294
294-295
295-296
296-297
297-298
298-299
299-300
300-301
301-302
302-303
303-304
304-305
305-306
306-307
307-308
308-309
309-310
310-311
311-312
312-313
313-314
314-315
315-316
316-317
317-318
318-319
319-320
320-321
321-322
322-323
323-324
324-325
325-326
326-327
327-328
328-329
329-330
330-331
331-332
332-333
333-334
334-335
335-336
336-337
337-338
338-339
339-340
340-341
341-342
342-343
343-344
344-345
345-346
346-347
347-348
348-349
349-350
350-351
351-352
352-353
353-354
354-355
355-356
356-357
357-358
358-359
359-360
360-361
361-362
362-363
363-364
364-365
365-366
366-367
367-368
368-369
369-370
370-371
371-372
372-373
373-374
374-375
375-376
376-377
377-378
378-379
379-380
380-381
381-382
382-383
383-384
384-385
385-386
386-387
387-388
388-389
389-390
390-391
391-392
392-393
393-394
394-395
395-396
396-397
397-398
398-399
399-400
400-401
401-402
402-403
403-404
404-405
405-406
406-407
407-408
408-409
409-410
410-411
411-412
412-413
413-414
414-415
415-416
416-417
417-418
418-419
419-420
420-421
421-422
422-423
423-424
424-425
425-426
426-427
427-428
428-429
429-430
430-431
431-432
432-433
433-434
434-435
435-436
436-437
437-438
438-439
439-440
440-441
441-442
442-443
443-444
444-445
445-446
446-447
447-448
448-449
449-450
450-451
451-452
452-453
453-454
454-455
455-456
456-457
457-458
458-459
459-460
460-461
461-462
462-463
463-464
464-465
465-466
466-467
467-468
468-469
469-470
470-471
471-472
472-473
473-474
474-475
475-476
476-477
477-478
478-479
479-480
480-481
481-482
482-483
483-484
484-485
485-486
486-487
487-488
488-489
489-490
490-491
491-492
492-493
493-494
494-495
495-496
496-497
497-498
498-499
499-500
500-501
501-502
502-503
503-504
504-505
505-506
506-507
507-508
508-509
509-510
510-511
511-512
512-513
513-514
514-515
515-516
516-517
517-518
518-519
519-520
520-521
521-522
522-523
523-524
524-525
525-526
526-527
527-528
528-529
529-530
530-531
531-532
532-533
533-534
534-535
535-536
536-537
537-538
538-539
539-540
540-541
541-542
542-543
543-544
544-545
545-546
546-547
547-548
548-549
549-550
550-551
551-552
552-553
553-554
554-555
555-556
556-557
557-558
558-559
559-560
560-561
561-562
562-563
563-564
564-565
565-566
566-567
567-568
568-569
569-570
570-571
571-572
572-573
573-574
574-575
575-576
576-577
577-578
578-579
579-580
580-581
581-582
582-583
583-584
584-585
585-586
586-587
587-588
588-589
589-590
590-591
591-592
592-593
593-594
594-595
595-596
596-597
597-598
598-599
599-600
600-601
601-602
602-603
603-604
604-605
605-606
606-607
607-608
608-609
609-610
610-611
611-612
612-613
613-614
614-615
615-616
616-617
617-618
618-619
619-620
620-621
621-622
622-623
623-624
624-625
625-626
626-627
627-628
628-629
629-630
630-631
631-632
632-633
633-634
634-635
635-636
636-637
637-638
638-639
639-640
640-641
641-642
642-643
643-644
644-645
645-646
646-647
647-648
648-649
649-650
650-651
651-652
652-653
653-654
654-655
655-656
656-657
657-658
658-659
659-660
660-661
661-662
662-663
663-664
664-665
665-666
666-667
667-668
668-669
669-670
670-671
671-672
672-673
673-674
674-675
675-676
676-677
677-678
678-679
679-680
680-681
681-682
682-683
683-684
684-685
685-686
686-687
687-688
688-689
689-690
690-691
691-692
692-693
693-694
694-695
695-696
696-697
697-698
698-699
699-700
700-701
701-702
702-703
703-704
704-705
705-706
706-707
707-708
708-709
709-710
710-711
711-712
712-713
713-714
714-715
715-716
716-717
717-718
718-719
719-720
720-721
721-722
722-723
723-724
724-725
725-726
726-727
727-728
728-729
729-730
730-731
731-732
732-733
733-734
734-735
735-736
736-737
737-738
738-739
739-740
740-741
741-742
742-743
743-744
744-745
745-746
746-747
747-748
748-749
749-750
750-751
751-752
752-753
753-754
754-755
755-756
756-757
757-758
758-759
759-760
760-761
761-762
762-763
763-764
764-765
765-766
766-767
767-768
768-769
769-770
770-771
771-772
772-773
773-774
774-775
775-776
776-777
777-778
778-779
779-780
780-781
781-782
782-783
783-784
784-785
785-786
786-787
787-788
788-789
789-790
790-791
791-792
792-793
793-794
794-795
795-796
796-797
797-798
798-799
799-800
800-801
801-802
802-803
803-804
804-805
805-806
806-807
807-808
808-809
809-810
810-811
811-812
812-813
813-814
814-815
815-816
816-817
817-818
818-819
819-820
820-821
821-822
822-823
823-824
824-825
825-826
826-827
827-828
828-829
829-830
830-831
831-832
832-833
833-834
834-835
835-836
836-837
837-838
838-

- R-p13 • Real-world performance,
or
R-p14 • physical prototyping platform

Final (A9)

- What can we do to increase speed.
- keep improving bottleneck
- lower image resolution
- accept lower frame rate.
- hardware changes etc
- Optimization - (A9)

1) debugging

- 2) can increase size of compiled code, which can be a problem in ENV with limited memory.
- 3) can increase computation time sometimes
- 4) not compatible for all hardware.

A4) So many parameters are converted into fixed/ constants. why?

- 1) increased performance
- 2) lower the memory usage
- 3) reduce memory fragmentation [dynamic]
- 4) easier debugging
- 5) less computation, less error possibility

* Real time embedded system tries to use less resources.

RTEs

- less flexibility of memory, processing power & i/o operations.

malloc → static

because we can not create new memory at run time.

* We change this in ~~HW~~ SW so that it matches HW situations

A5) After increasing the number of pixels, the stack size would become larger

- ↳ Variables / array ↑
- * which get allocated on a stack.

pipeline

A8) instead of waiting for the blur op to complete on 1 image before starting the next op, we can pipeline and start the next op while the current one is still being processed

changes in delay

When profiling an application with the GNU profiler on a shared server, the results can vary due to users running heavy application.

because CPU, memory usage are shared among multiple users & applications running concurrently on the server.

heavy applications running will consume large amount of resources leading to other application being profiled.