6.5081: Interrupts

$15

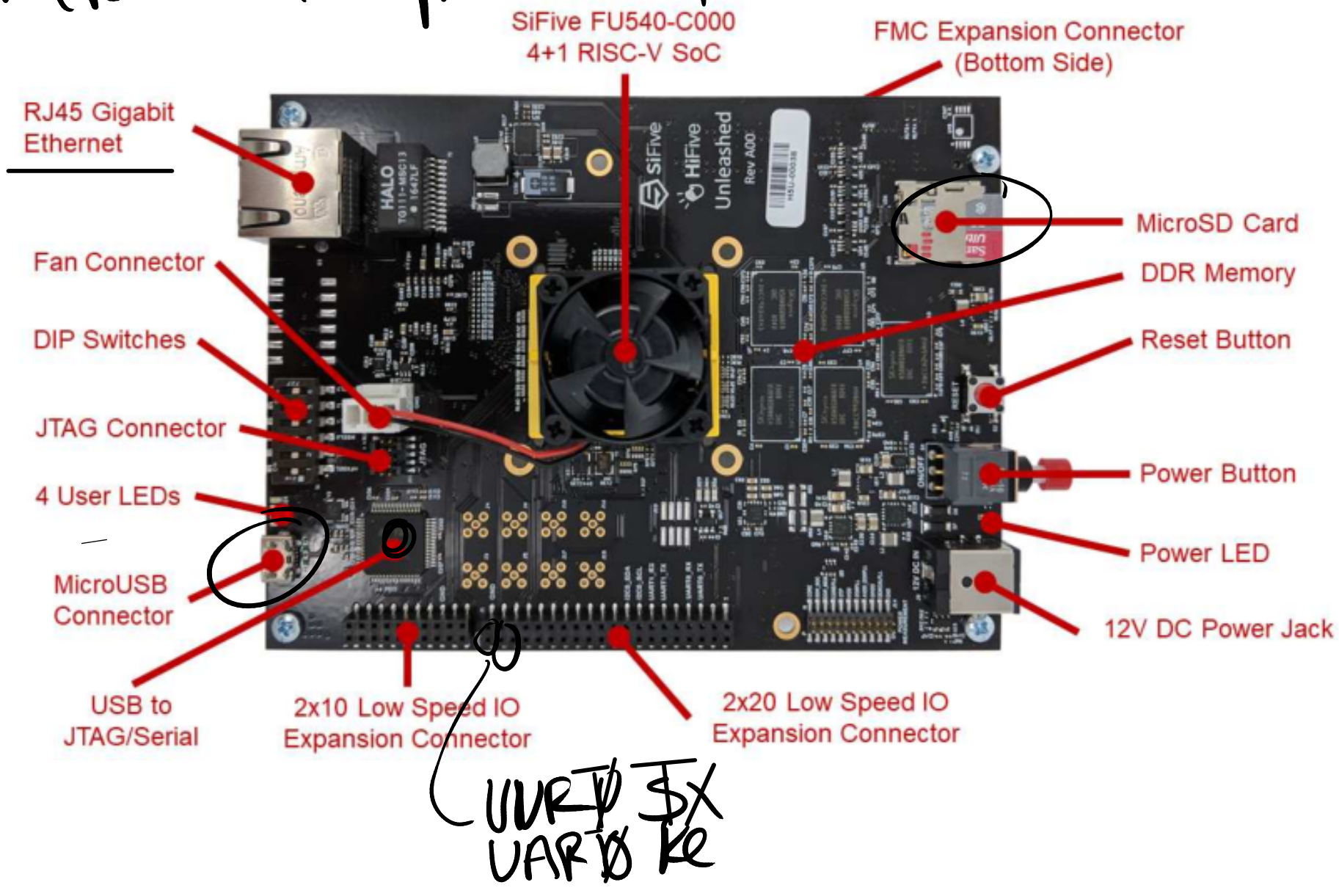
HW wants attention now

SW save its work
process interrupt
resume its work

for syscall + traps they all use same mechanism

1) asynchronous
2) concurrency
3) program devices

Where do interrupts come from?



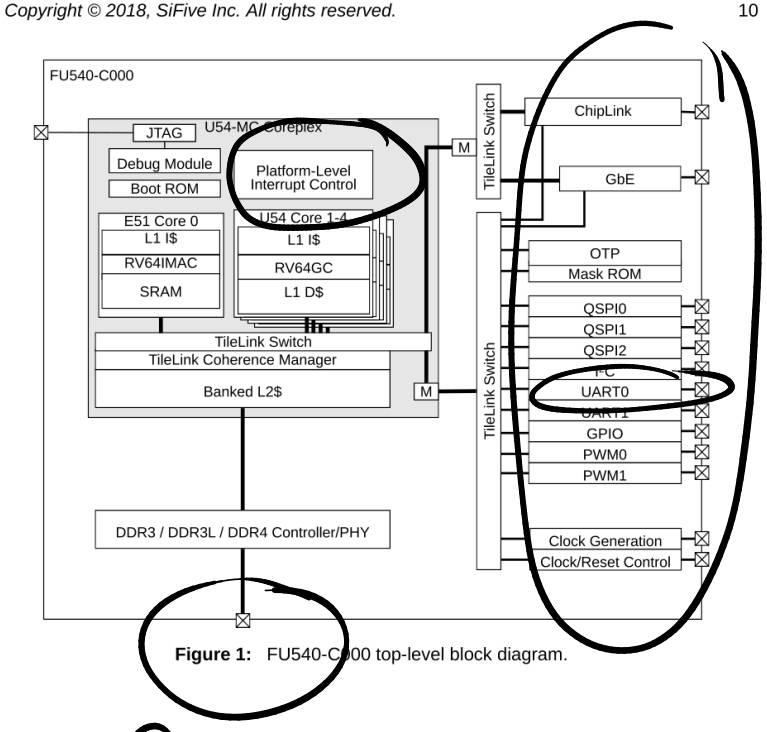


Figure 1: FU540-C000 top-level block diagram.

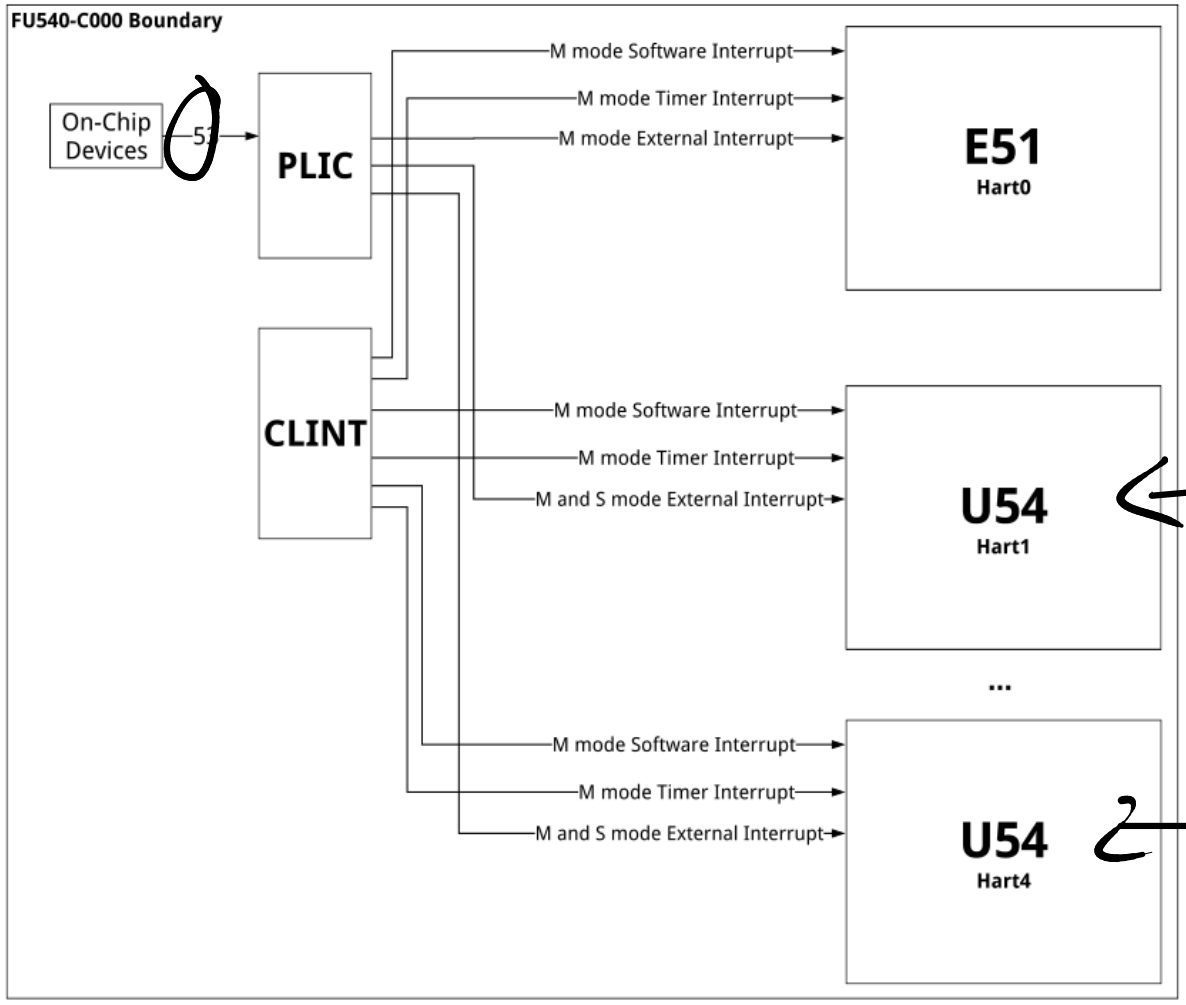
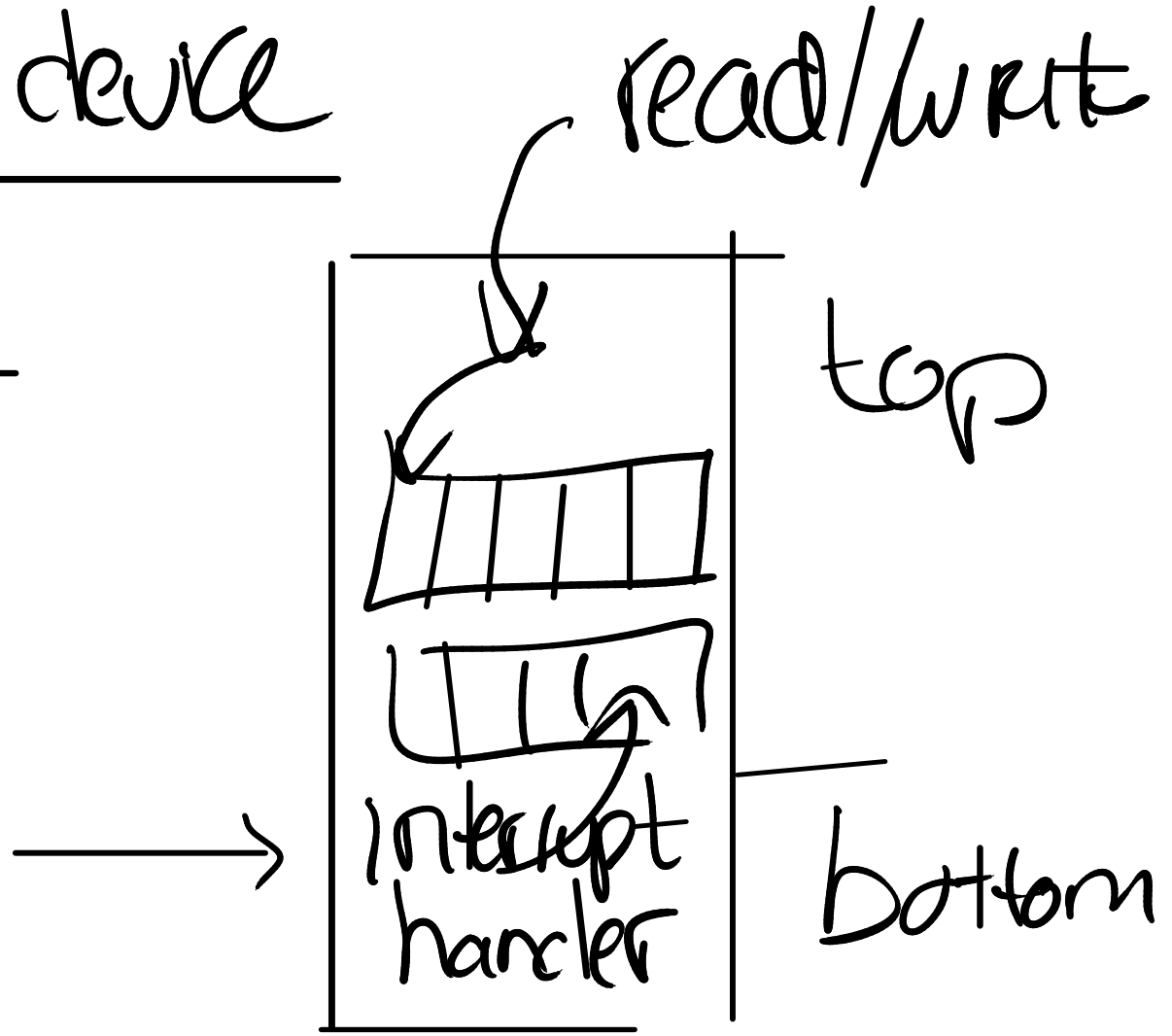


Figure 3: FU540-C000 Interrupt Architecture Block Diagram.

Driver manages device

UART.c



Programming device

memory mapped i/o / ld/st

ld/st read/write control registers of the device

Case Study: $ ls

$: device puts $ into uart
uart gen interrupt when the
char has been sent

ls: keyboard control to recv line
generate interrupt

RISC-V support for interrupts

SIE: one bit for E, S, T

SSTATUS: bit enable/disable

SIP: interrupt pending

scause:

STVEC:

Interrupt (hw)

If SIE bit set:

clear SIE bit

sepc ← pc

save current mode

mode ← supervisor

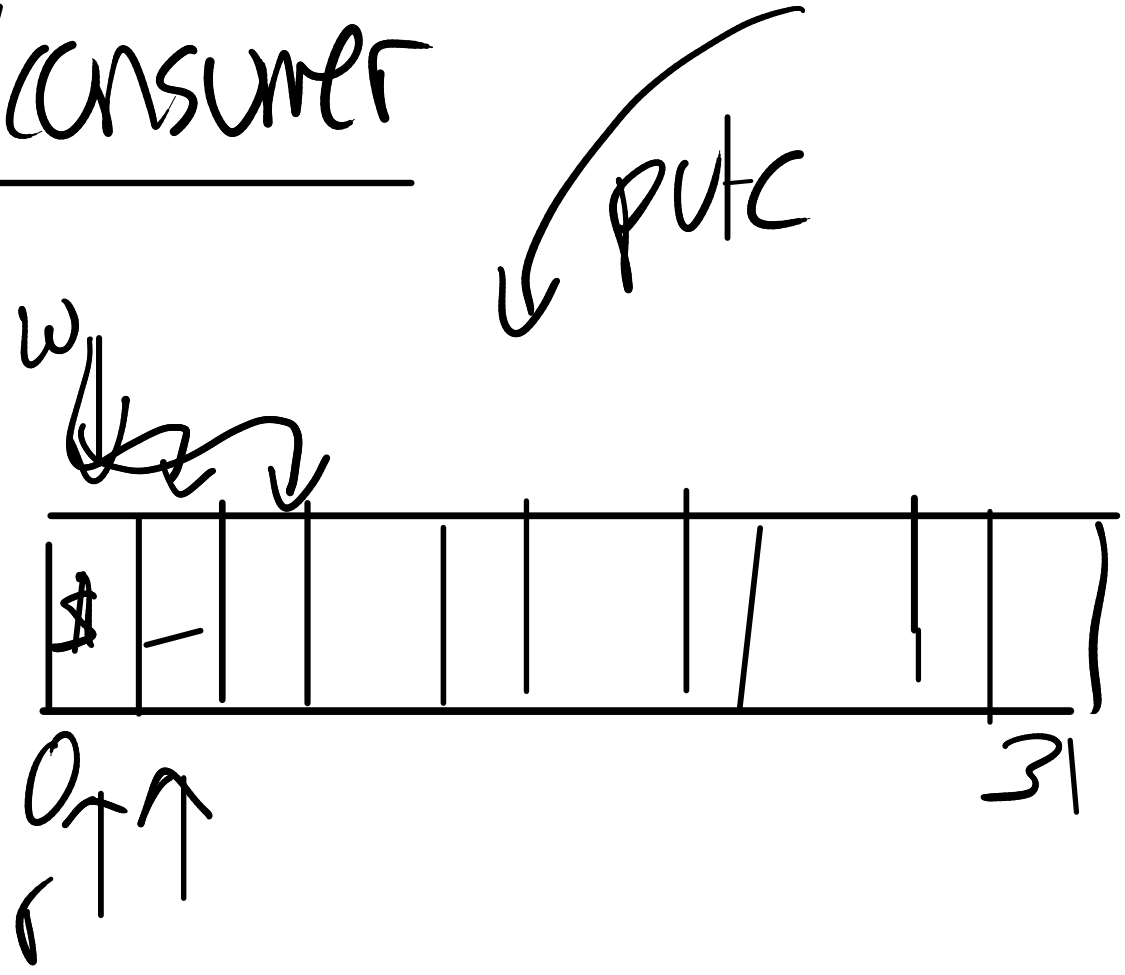
pc ← stvec

↳ usertrap()

Interrupts and concurrency

1) device + CPU run in parallel
   ⇒ producer/consumer parallelism
2) interrupt stops the current program
   interrupt enable/disable
3) top of driver + bottom driver may run in parallel using locks ⇒ lock

Producer/consumer



UARTintr()

Interrupt evolution

Interrupt used to be fast
simpl

Now slow
device is more complicated
Gbit ethernet
1.5 mpkt/s ⇒ 1 interrupt per μsec

Polling

CPU spins until device has data

Waste CPU cycles if device is slow

but if device is fast,

saves entry/exit cost

Dynamically switch between polling/interrupts