

# HLS DUTh Lab

ΣΑΜΟΛΑΔΑΣ ΤΡΙΑΝΤΑΦΥΛΛΟΣ 57259

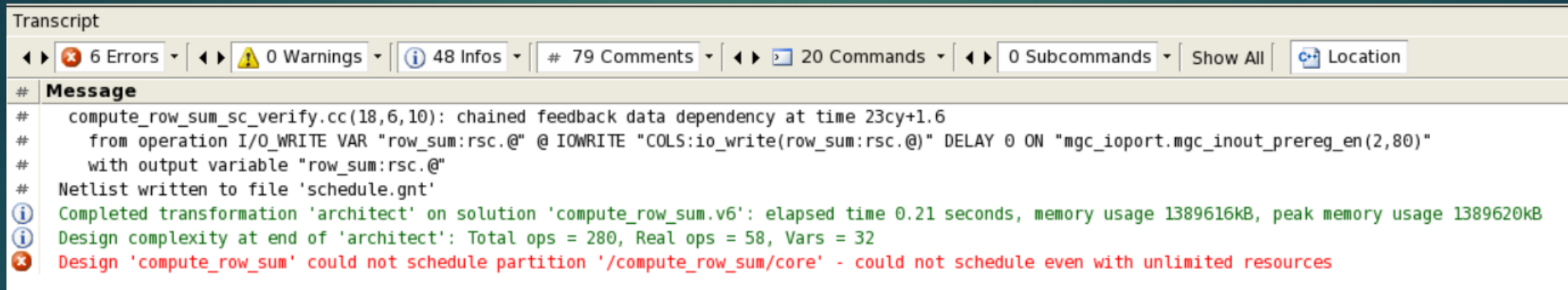
# HLS DUTh Lab

| Table   Constraint Editor   Schedule   RTL Schematic   compute row sum sc ve... |                |              |           |           |             |       |
|---|----------------|--------------|-----------|-----------|-------------|-------|
| Report: General   |                |              |           |           |             |       |
| Solution  | Latency Cycles | Latency Time | Throug... | Throug... | Total Ar... | Slack |
| compute_row_sum.v1 (extract)  | 42             | 84.00        | 47        | 94.00     | 921.00      | 0.07  |
| compute_row_sum.v2 (allocate)   | 42             | 84.00        | 47        | 94.00     | 1704.18     |       |
| compute_row_sum.v3 (allocate)   | 32             | 64.00        | 37        | 74.00     | 1704.18     |       |
| compute_row_sum.v4 (allocate)   |                |              |           |           |             |       |
| compute_row_sum.v5 (extract)  | 29             | 58.00        | 33        | 66.00     | 1102.82     | 0.50  |

Στον παραπάνω πίνακα παρουσιάζονται διάφορες δοκιμές που έγιναν στο Catapult.

- Στην v3 εφαρμόζουμε loop pipeline με  $ll=1$  μόνο στην εσωτερική COLS loop.
- Στην v5 εφαρμόζουμε loop pipeline με  $ll=2$  στην εξωτερική ROW loop (άρα και στην εσωτερική) ενώ για την διεπαφή μνημών χρησιμοποιούμε τις dualport μνήμες. Το  $ll=2$  είναι το **ελάχιστο** initiation interval που μπορεί να επιτευχθεί χωρίς να γίνει κάποια τροποποίηση στον κώδικα, και ο περιορισμός μας σε αυτό είναι ο αριθμός των resources

# HLS DUTh Lab










The screenshot shows the 'Transcript' window in the Xilinx Vivado IDE. The window has a toolbar with icons for navigating through messages: 6 Errors (red X), 0 Warnings (yellow triangle), 48 Infos (blue i), 79 Comments (hash), 20 Commands (blue square), and 0 Subcommands (blue square). There are also buttons for 'Show All' and 'Location'. The message list is as follows:

```
# Message
# compute_row_sum_sc_verify.cc(18,6,10): chained feedback data dependency at time 23cy+1.6
#   from operation I/O_WRITE VAR "row_sum:rsc.@" @ IOWRITE "COLS:io_write(row_sum:rsc.@" DELAY 0 ON "mgc_ioport.mgc_inout_prereg_en(2,80)"
#   with output variable "row_sum:rsc.@"
# Netlist written to file 'schedule.gnt'
(i) Completed transformation 'architect' on solution 'compute_row_sum.v6': elapsed time 0.21 seconds, memory usage 1389616kB, peak memory usage 1389620kB
(i) Design complexity at end of 'architect': Total ops = 280, Real ops = 58, Vars = 32
(x) Design 'compute_row_sum' could not schedule partition '/compute_row_sum/core' - could not schedule even with unlimited resources
```

Η ν5 που είδαμε προηγουμένως ότι απέτυχε ήταν προσπάθεια για  $ll=1$  στην ROW loop και όπως φαίνεται στην παραπάνω εικόνα ο λόγος αποτυχίας είναι τα resources.

# HLS DUTh Lab

| Solution ^   | Latency... | Latency...   | Throug... | Throug...    | Total Ar...    | Slack |
|--|------------|--------------|-----------|--------------|----------------|-------|
|  compute_row_sum.v1 (extract)         | 29         | 58.00        | 33        | 66.00        | 750.65         | 0.26  |
|  solution.v1 (analyze)                |            |              |           |              |                |       |
|  compute_row_sum.v2 (extract)         | 30         | 60.00        | 33        | 66.00        | 1644.33        | 0.25  |
|  compute_row_sum.v3 (allocate)        | 16         | 32.00        | 19        | 38.00        | 2237.00        |       |
|  compute_row_sum.v4 (allocate)        | 16         | 32.00        | 15        | 30.00        | 1756.68        |       |
|  compute_row_sum.v5 (allocate)        | 16         | 32.00        | 15        | 30.00        | 1756.68        |       |
|  <b>compute_row_sum.v6 (allocate)</b> | <b>15</b>  | <b>30.00</b> | <b>15</b> | <b>30.00</b> | <b>1919.87</b> |       |

Για να επιτύχουμε το  $ll=1$ , αλλάζουμε τον κώδικα μας προσθέτοντας μια τοπική μεταβλητή `temp` σε ρόλο accumulator, πάνω στην οποία προσθέτουμε συνεχώς τις τιμές του πίνακα `a`. Αφού υπολογίσουμε το άθροισμα της σειράς, τότε το καταχωρούμε στην `row_sum[i]`. Έτσι μειώνουμε σημαντικά τον αριθμό των memory access(read/write) που γίνονται σε κάθε loop ROW. Όπως φαίνεται, το latency πέφτει μόλις στους 15 κύκλους με latency time=30.