Virtualbox VM for Bluespec Development

Sang-Woo Jun

2022

UCI

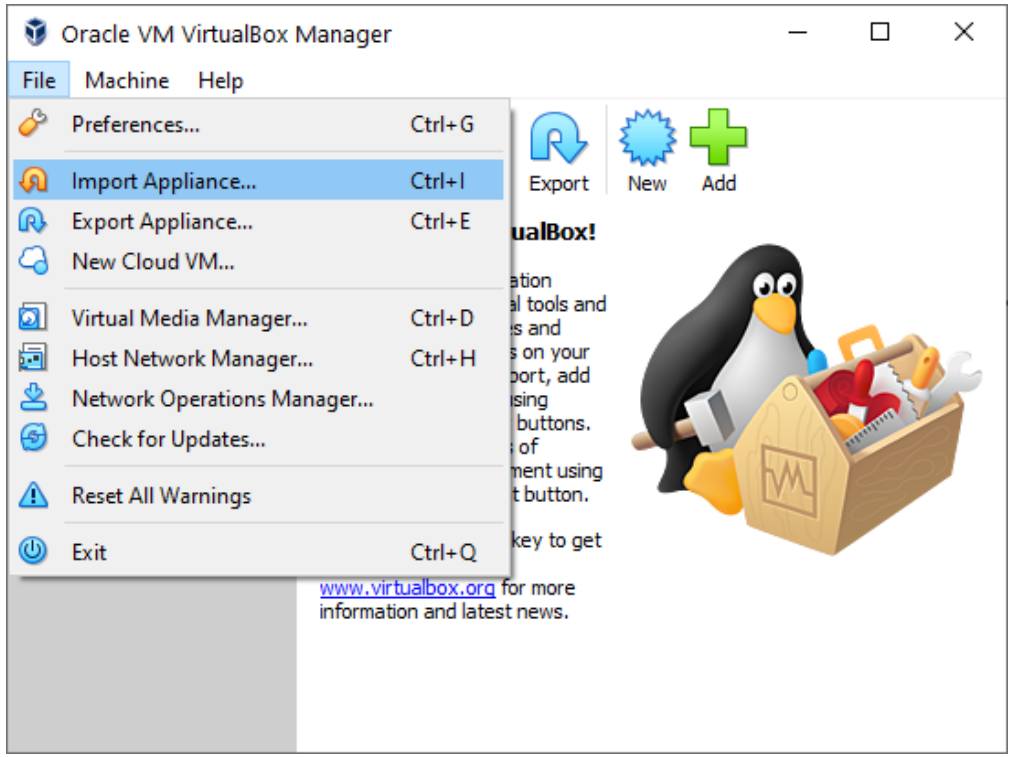
Getting Started

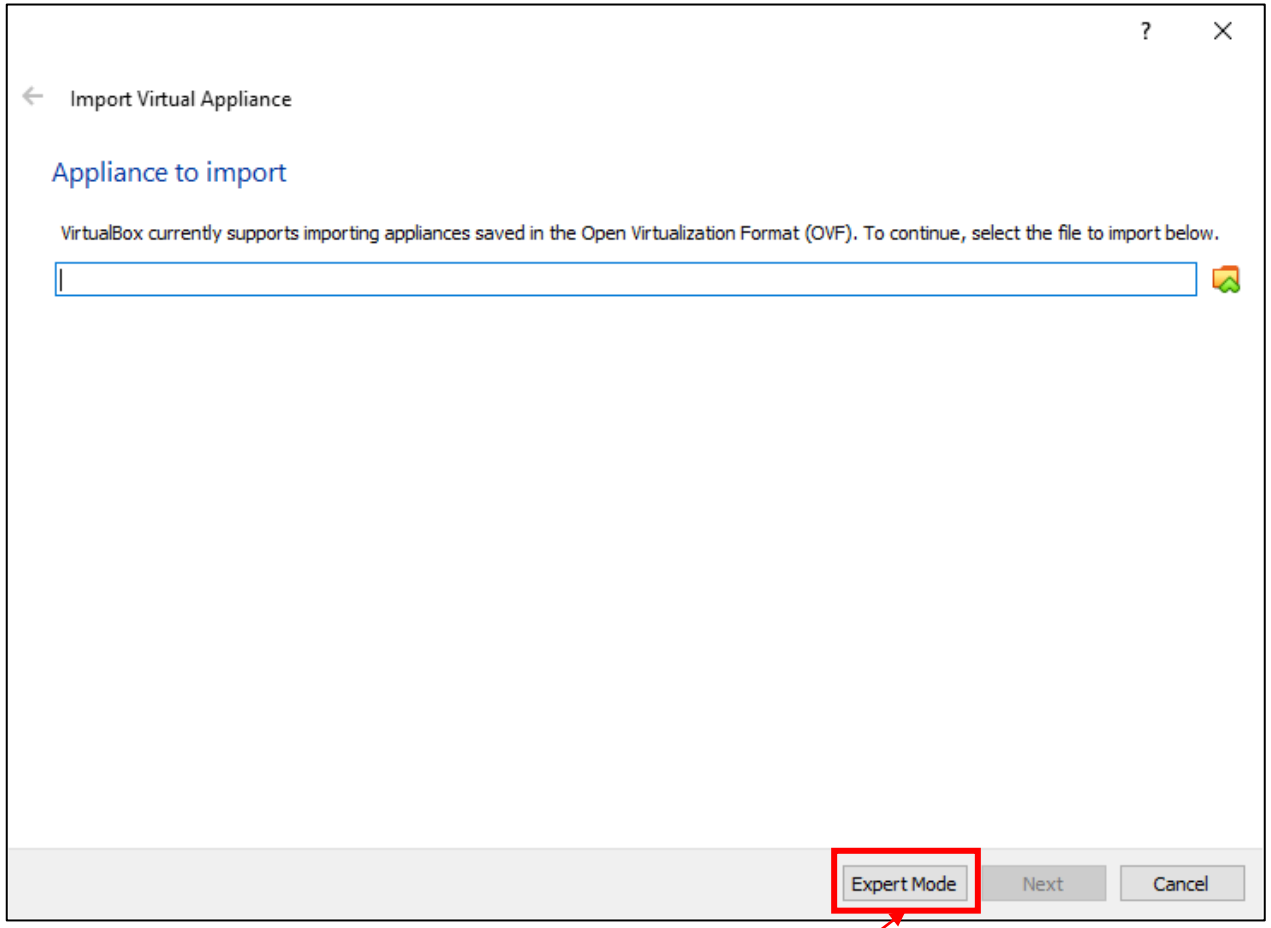
- Virtual machine with all tools installed, available at:
  - cs152-ubuntu.ova (4 GB!)
  - Created for CS152 : Undergraduate computer systems architecture

    https://drive.google.com/file/d/1plT9o1QIeDkci0l_jB4Si9BTwmqmeIgF/view
- First, install Oracle Virtualbox
  - Open-source virtual machine
  - High performance with minimal configuration

Getting Started

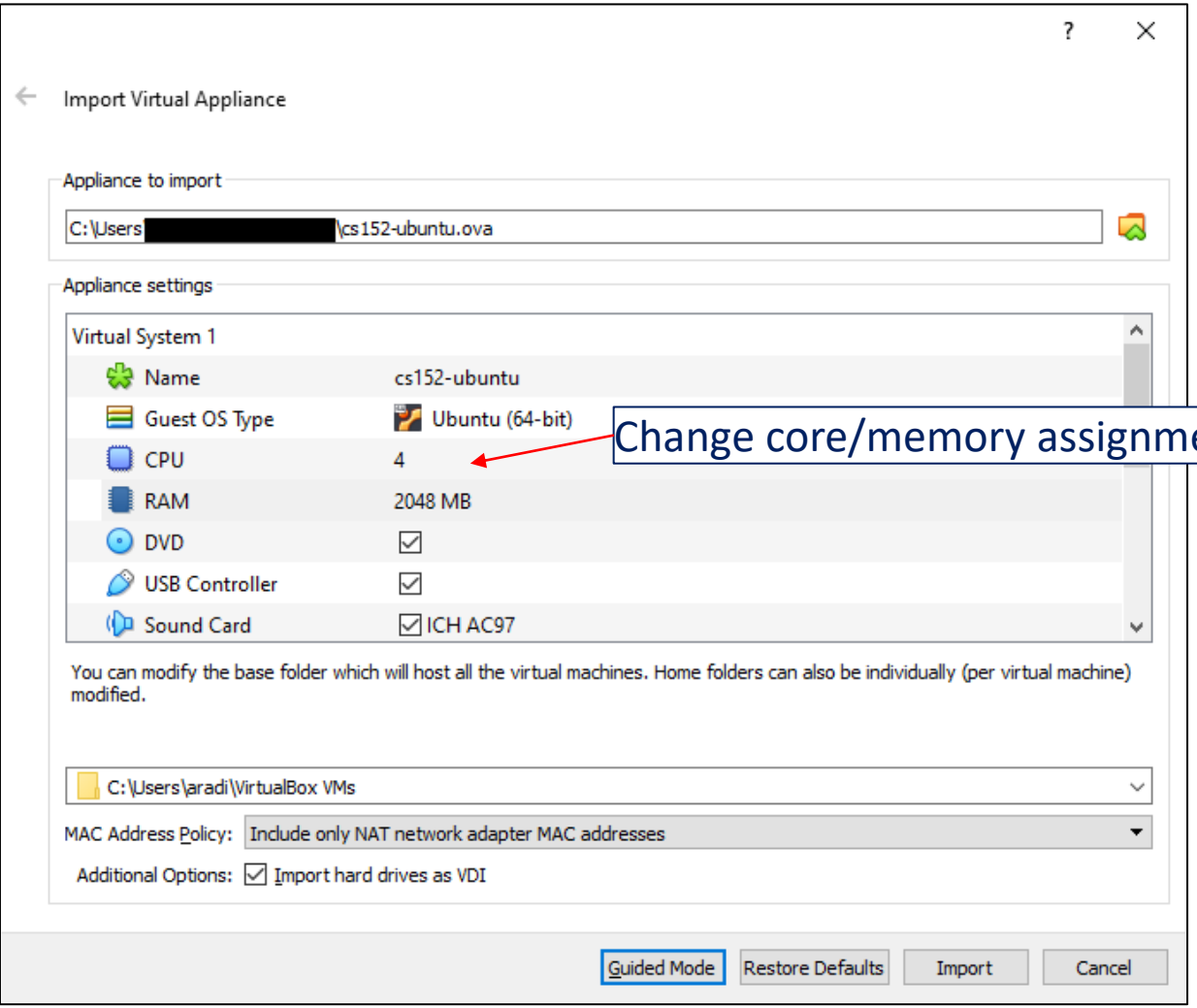
- Import the downloaded VM



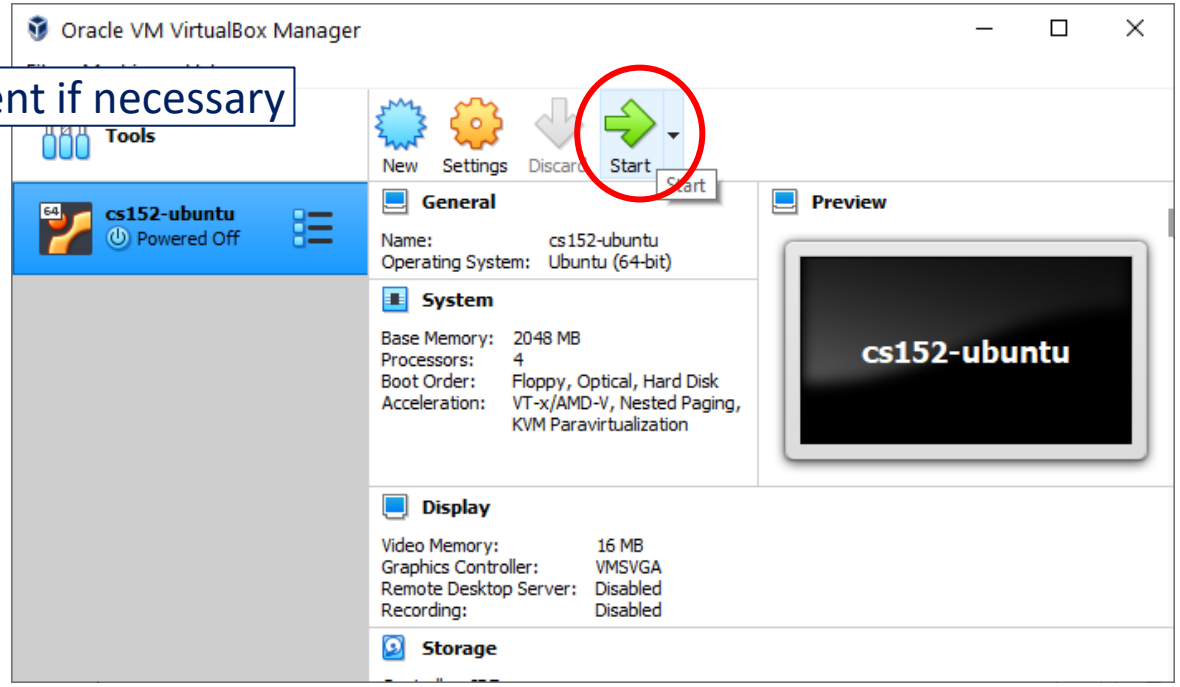


If core count/memory allowance needs changing

Getting started



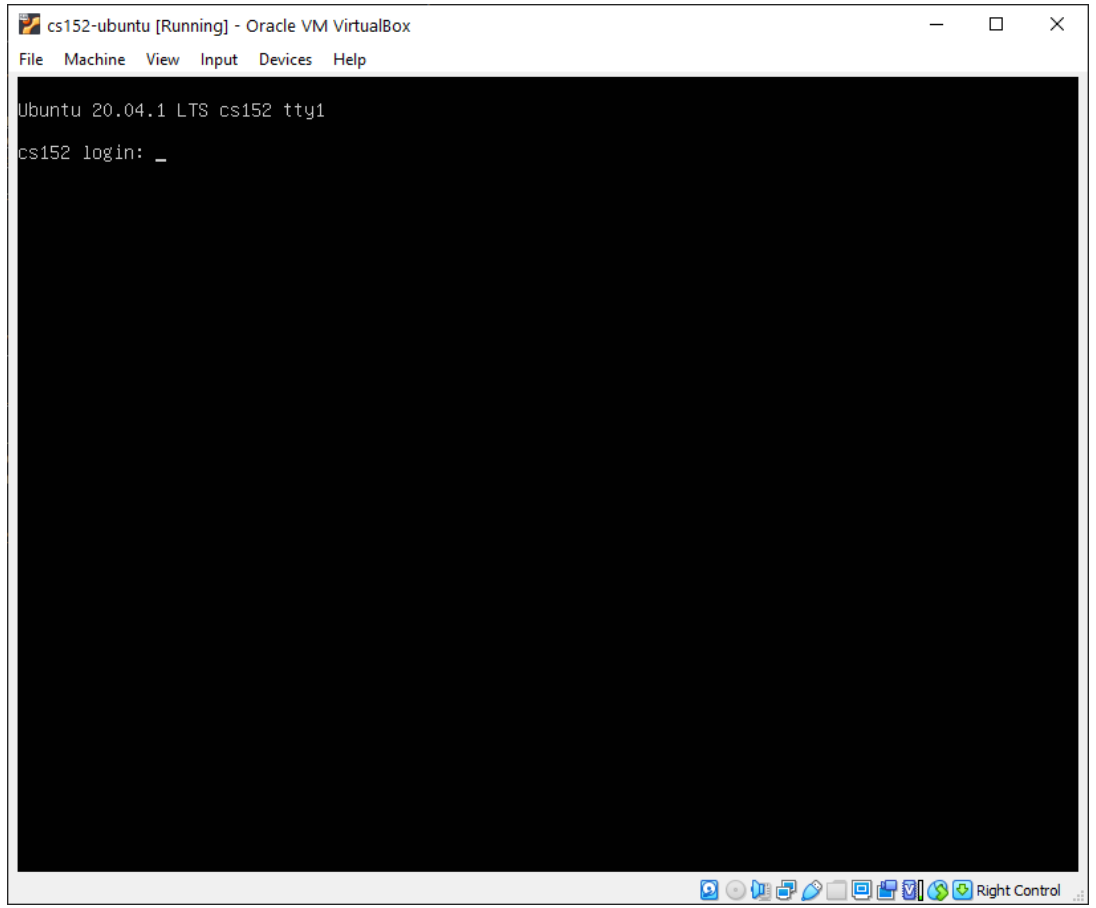
Change core/memory assignment if necessary



Getting started

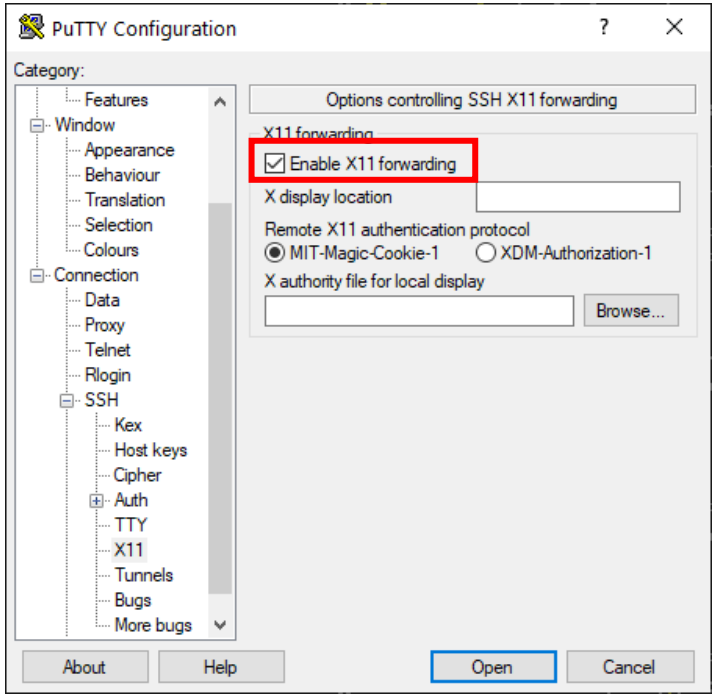
- You can work in the VM window, OR
- Connect to it via a terminal
  - Putty, MobaXterm, OpenSSH, etc
- The VM forwards its
  - port 22 (ssh) to
  - 3022
  - Connect to it by ssh cs152@127.0.0.1:3022
- Login: cs152/cs152
- Run ./clone-ulx3s.sh

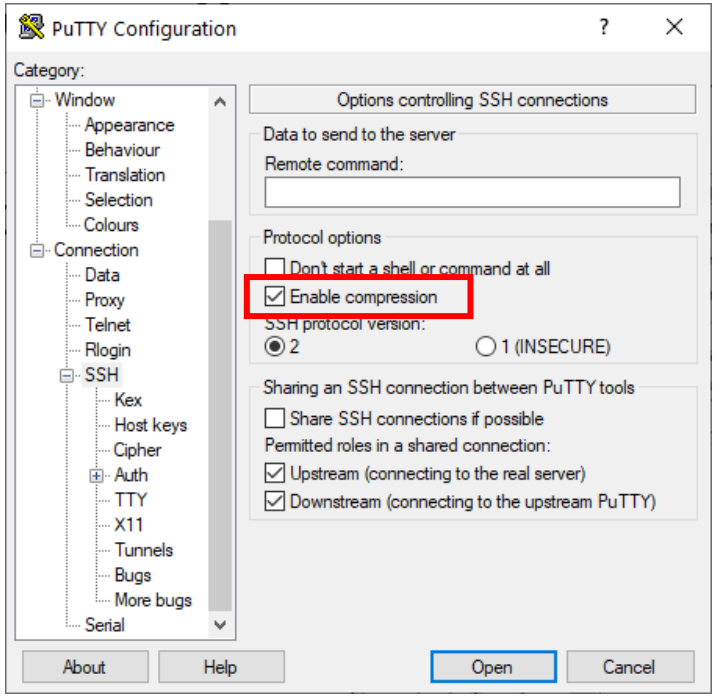
Check it out!



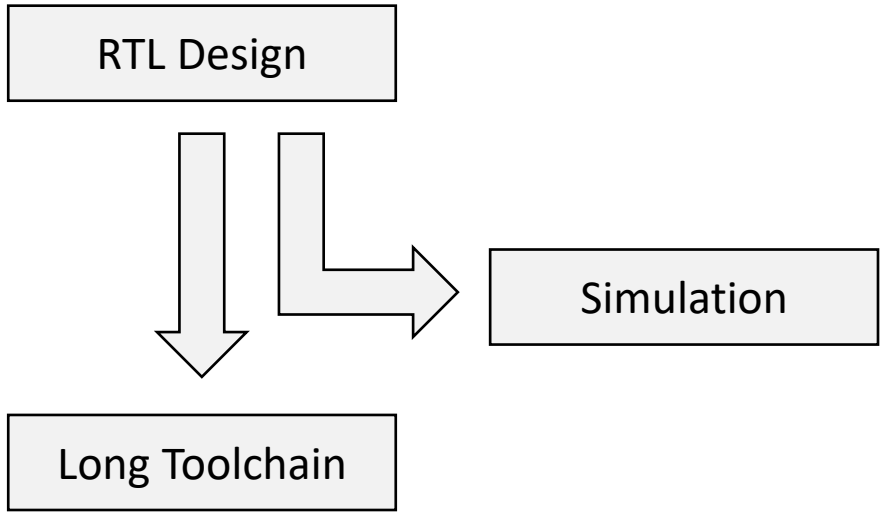
X Forwarding for GUI

- For Windows
  - Install and run an X-server (Xming, MobaXterm, …)
  - Enable X forwarding (+compression for performance) in your SSH client
- For Linux
  - ssh –XC 127.0.0.1:3022





Trying simulation - example



- cs152-rv32i-bsv/projects/rv32i/
- Compiling and running the simulation
  - “make bsim” – Stands for “bluesim”
  - “make runsim” creates two files
    - system.log : log of processor operation
    - output.log : log of software output
- Default benchmark: Sudoku solver
  - Source: sw/minisudoku.c
  - Resulting assembly: sw/minisudoku.dump
  - Binary for processor: sw/minisudoku.bin

```
155 0000023c <solve>:
156  23c:→  fd010113          → addi→   sp,sp,-48
157  240:→  02112623          → sw→ ra,44(sp)
158  244:→  02812423          → sw→ s0,40(sp)
159  248:→  03010413          → addi→   s0,sp,48
160  24c:→  fca42e23          → sw→ a0,-36(s0)
161  250:→  fcb42c23          → sw→ a1,-40(s0)
162  254:→  fd842703          → lw→ a4,-40(s0)
163  258:→  00f00793          → addi→   a5,zero,15
164  25c:→  00e7d663          → bge→a5,a4,268 <solve+0x2c>
```

Default hardware target platform

- Lattice ECP5-85F FPGA
- Host software loads software/data over USB to FPGA
- Host software communicates with FPGA via UART over USB

Synthesizing a bitfile - example

- "make" creates a bitfile, and prints a lot of logs onto screen
  - "make | tee build.log" to analyze output
  - Log file is long!
- Example log files from synthesis:
  - Look for "Device utilisation" [sic]:

```
Info: Device utilisation:
Info: →         TRELLIS_SLICE:  4982/41820    11%
```

  - Look for "Max frequency" :

```
Info: Max frequency for clock '$glbnet$CLK_clk_25mhz$TRELLIS_IO_IN': 69.80 MHz (PASS at 25.00 MHz)
```

  - Look for "Critical path report for clock":

```
Info: Critical path report for clock '$glbnet$CLK_clk_25mhz$TRELLIS_IO_IN' (posedge -> posedge):
Info: curr total
Info:  0.5  0.5  Source main_proc.imemRespQ.data0_reg_TRELLIS_FF_Q_30_DI_PFUMX_Z_SLICE.Q0
Info:  1.5  2.0    Net main_proc.imemRespQ_D_OUT[1] budget 5.041000 ns (33,27) -> (33,28)
```

Where is the critical path?

- Look at the synthesis log!

```
Info: Critical path report for clock '$glbnet$CLK_clk_25mhz$TRELLIS_IO_IN' (posedge -> posedge):
Info: curr total
Info:  0.5  0.5  Source main_proc.imemRespQ.data0_reg_TRELLIS_FF_Q_30_DI_PFUMX_Z_SLICE.Q0
Info:  1.2  1.7    Net main_proc.imemRespQ_D_OUT[1] budget 3.042000 ns (44,26) -> (43,27)
                                                  ⋮
Info:  0.2 14.2  Source main_proc.d2e.data0_reg_TRELLIS_FF_Q_108_DI_L6MUX21_Z_D1_L6MUX21_Z_D0_PFUMX_Z_SLICE.OFX1
Info:  0.1 14.3    Net main_proc.d2e.data0_reg_TRELLIS_FF_Q_108_DI budget 5.039000 ns (8,40) -> (8,40)
Info:                Sink main_proc.d2e.data0_reg_TRELLIS_FF_Q_108_DI_L6MUX21_Z_D1_L6MUX21_Z_D0_PFUMX_Z_SLICE.DI1
Info:  0.0 14.3  Setup main_proc.d2e.data0_reg_TRELLIS_FF_Q_108_DI_L6MUX21_Z_D1_L6MUX21_Z_D0_PFUMX_Z_SLICE.DI1
Info: 3.8 ns logic, 10.5 ns routing
```