DESIGN AND TUNING OF A TREE-MESH CLOCK DISTRIBUTION

Nikhil Jayakumar, Dave Murata, Valery Kugel
{ nikhilj, dmurata , valery } @juniper.net
Juniper Networks

OVERVIEW

- Comparison of Clock trees vs Clock Grids/Mesh

- Juniper's Clock distribution design overview

- Juniper's 2 step tuning flow for clock meshes
  - Coarse Tuning
  - Fine Tuning

- Conclusion

CLOCK TREES VS CLOCK GRIDS

There are 2 two kinds of clock skews

- Structural (layout) skew
  - Capacitive load mismatch
  - Wire length mismatch

- Skew due to PVT variations

STA ✓

Handled by balanced clock trees (Eg: Htrees)
- Zero / low skew only in the absence of PVT variations
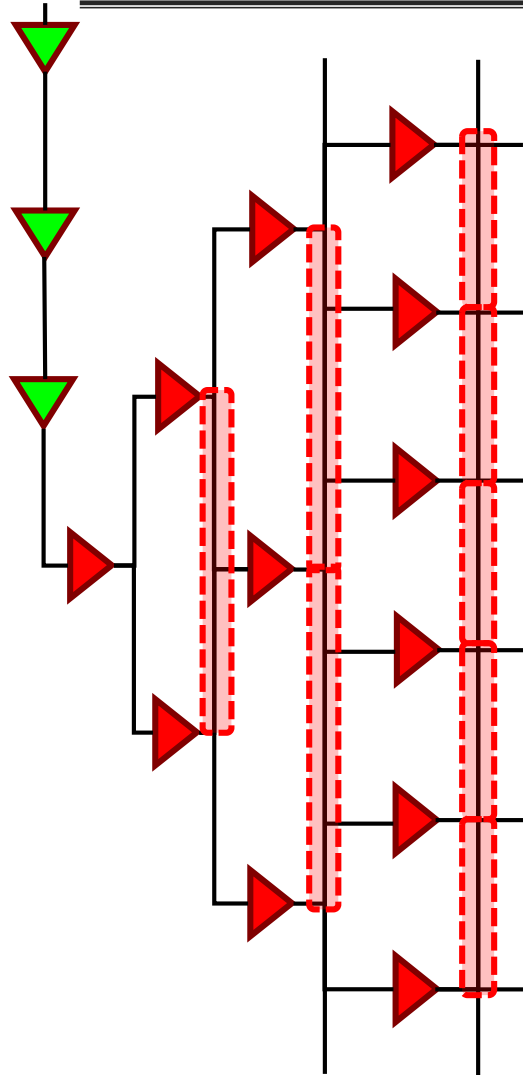
Two types of approaches to handle this:
- Dynamic:
  - Dynamic clock deskewing schemes
- Static:
  - Cross-link addition
  - Clock mesh / grid / hybrid tree-mesh

Necessitates SPICE based analysis
- Regular STA won't work due to re-convergences. (more on this later....)

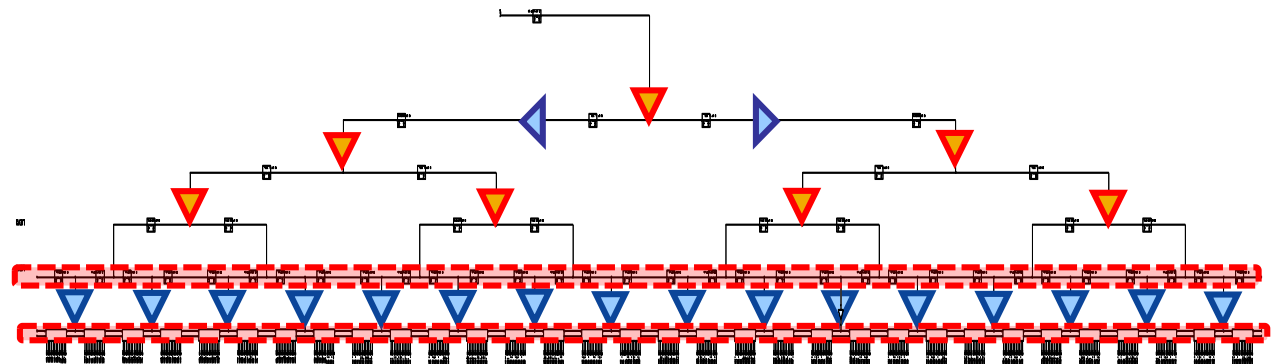# VERTICAL CLOCK SPINE + HORIZONTAL CLOCK RIBS

Constructed to
- be balanced
- have low latency (and hence low jitter)

Wire width, spacing, buffer drive strength, wire length between buffers chosen after careful simulation. Factors considered:
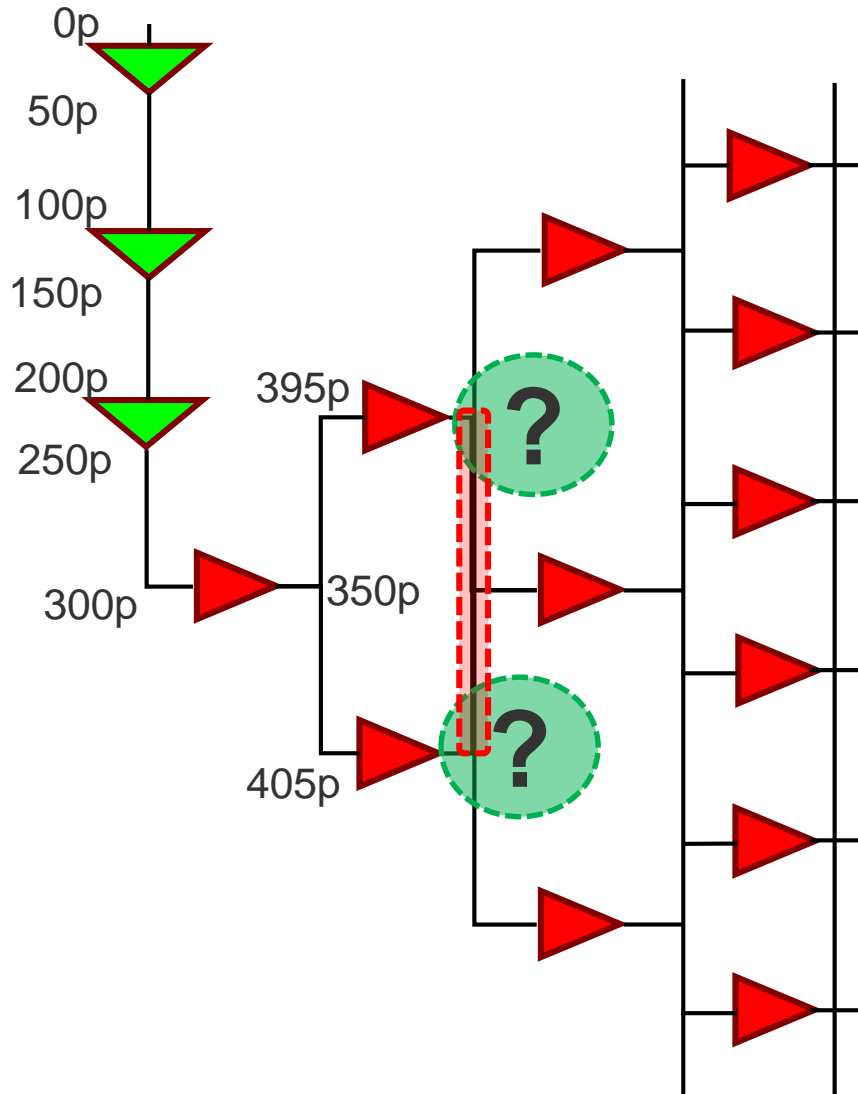- Jitter (chose wire code for minimum jitter per unit length)
- Slew constraints
- Dynamic IR drop & EM limits
- Routability &area constraints
- Overshoot & undershoot due to inductance

Cancel out PVT variations through insertion of cross-links (shorting wires) at regular intervals.
- Cross-links were inserted only if skew reduction outweighed jitter increase.

WHY CROSS-LINKS COMPLICATE TIMING?

STA cannot handle re-convergence in non-linear circuits.

- SPICE confirms the averaging effect of the short, but STA cannot see this.
- Where is the point of divergence?

Need a SPICE simulation to estimate delays.

JUNIPER GLOBAL CLOCK DISTRIBUTION
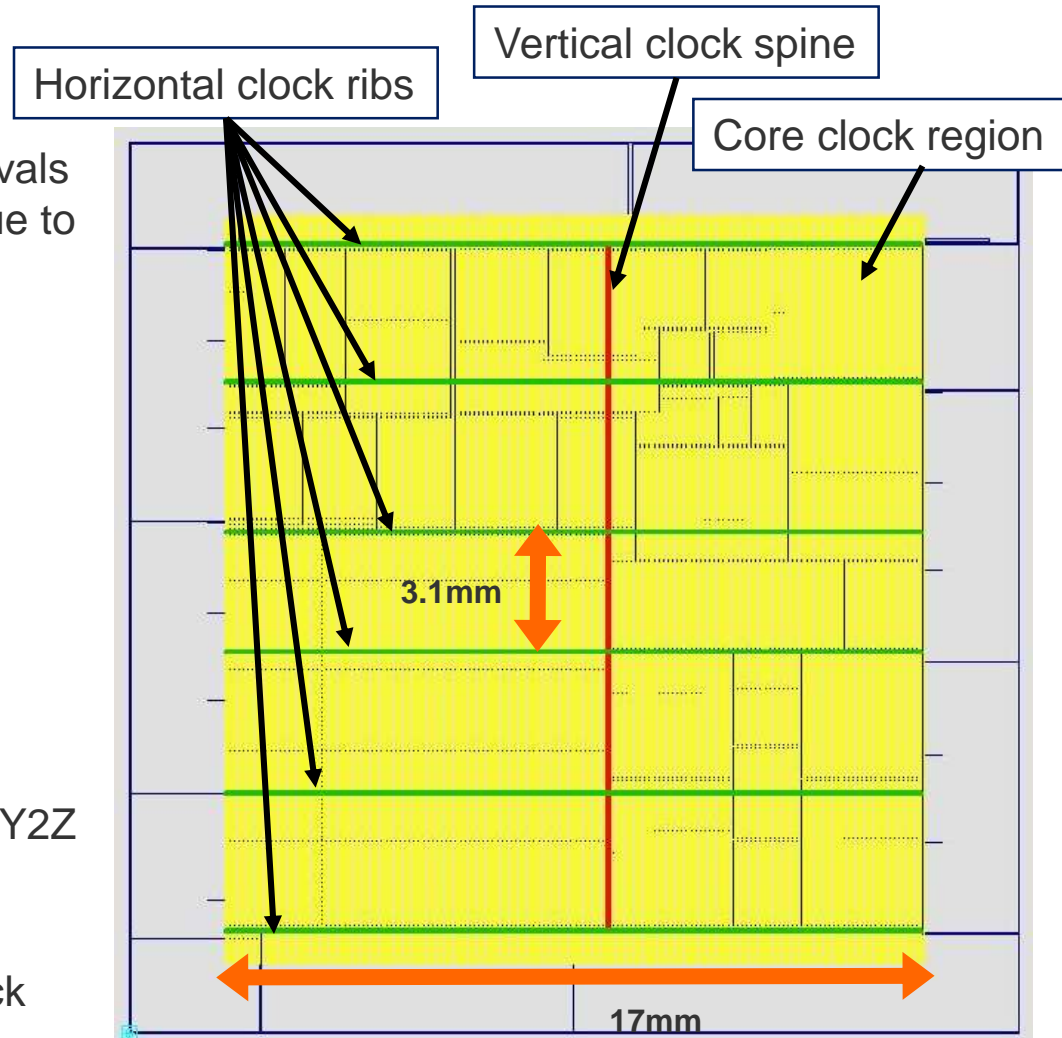
Hybrid tree-mesh

- Balanced tree driving a mesh
- Cross-links added at regular intervals in the tree also to reduce skew due to PVT

Construction:

- PLL drives Vertical Spine
- Vertical Spine drives 6 Horizontal Ribs
- Horizontal Ribs drive clock mesh

Technology Details:

- Frequency: 700Mhz to 800Mhz
- TSMC 40nm (45GS_1P10M_6X1Y2Z + Al RDL)
  - Top 2 (thick) metal layers (Mz) used to distribute the core clock

Horizontal clock ribs

Vertical clock spine

Core clock region

3.1mm

17mm

WHY REDUCE SKEW IN A MESH?

Q : Clock meshes reduce skew - so then why do we have to tune it?

- Clock meshes have an effect of averaging the delay – but at the cost of short circcuit current
- Large skew can result in a very large short-circuit current for drivers whose outputs are shorted

Should not rely on the mesh to reduce structural skew. The mesh is used to only reduce PVT skew.
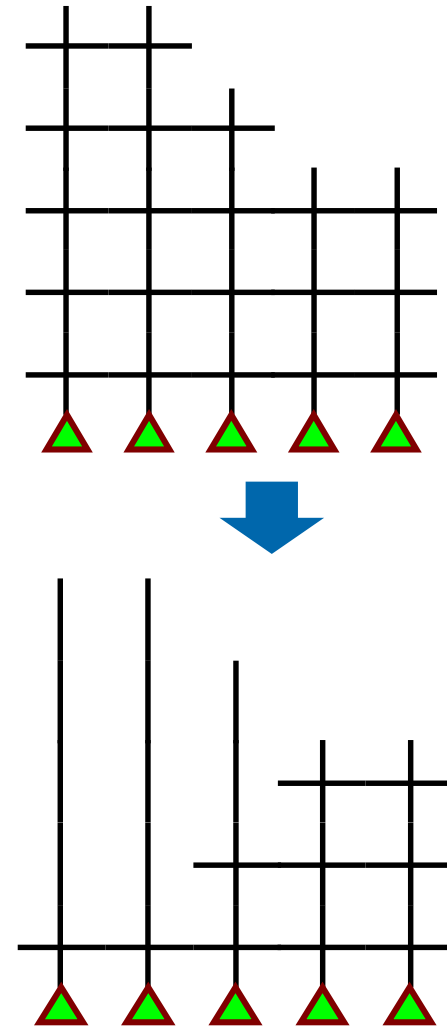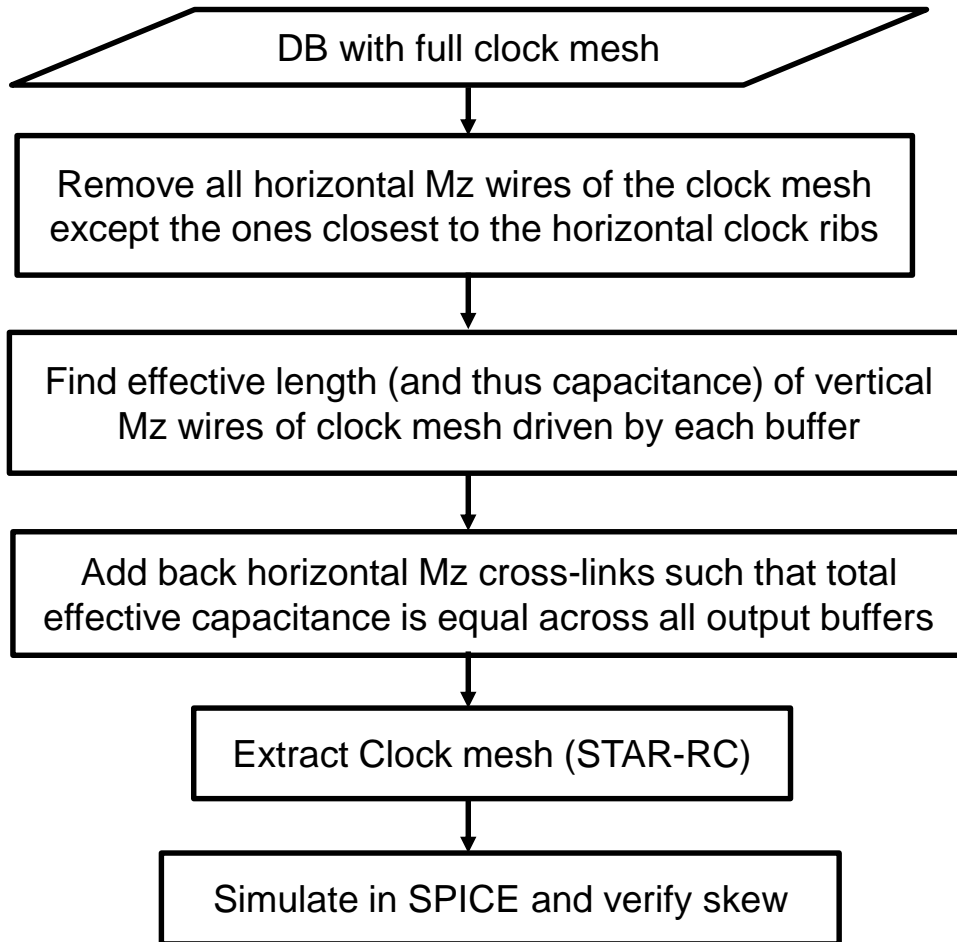
JUNIPER'S 2 STEP TUNING FLOW

1. Coarse-tuning through balancing

   - Tuning the vertical spine and horizontal ribs through RC balancing
   - Tuning the mesh through selective removal of horizontal cross-link wires in the mesh
     - Based on effective wire length (capacitance) driven by each buffer
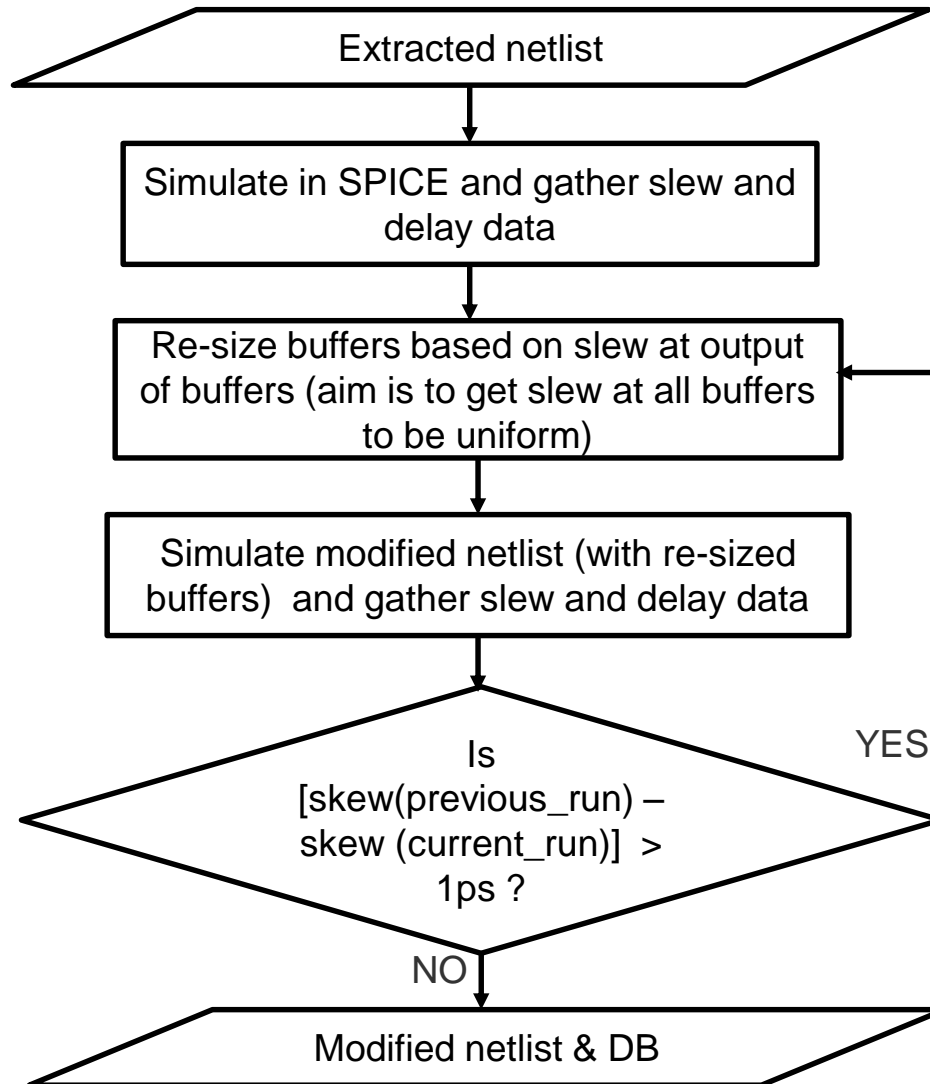
2. Fine-tuning through driver sizing

   - Automatic driver tuning flow that sizes drivers in the vertical spine and horizontal ribs
     - Drivers are sized to achieve uniform output delay and slew
   - Flow can simultaneously size several thousands of buffers
     - Manual tuning is impossible on such a scale

COARSE TUNING FLOW OF THE MESH

DB with full clock mesh

↓

Remove all horizontal Mz wires of the clock mesh except the ones closest to the horizontal clock ribs

↓

Find effective length (and thus capacitance) of vertical Mz wires of clock mesh driven by each buffer

↓

Add back horizontal Mz cross-links such that total effective capacitance is equal across all output buffers

↓

Extract Clock mesh (STAR-RC)

↓

Simulate in SPICE and verify skew

FINE TUNING FLOW

Extracted netlist

Simulate in SPICE and gather slew and delay data

Re-size buffers based on slew at output of buffers (aim is to get slew at all buffers to be uniform)

Simulate modified netlist (with re-sized buffers) and gather slew and delay data

Is [skew(previous_run) – skew (current_run)] > 1ps ?

YES

NO

Modified netlist & DB

Buffers are sized based on output slew

- If slew is larger than target slew, the buffer is up-sized proportionally to achieve target slew

- If slew is smaller than target slew, the buffer is down-sized proportionally to achieve target slew

The fine tuning flow is able to converge to a low-skew solution within 2 to 3 iterations

Buffers can be re-sized without re-extracting since the buffers are designed to be footprint compatible

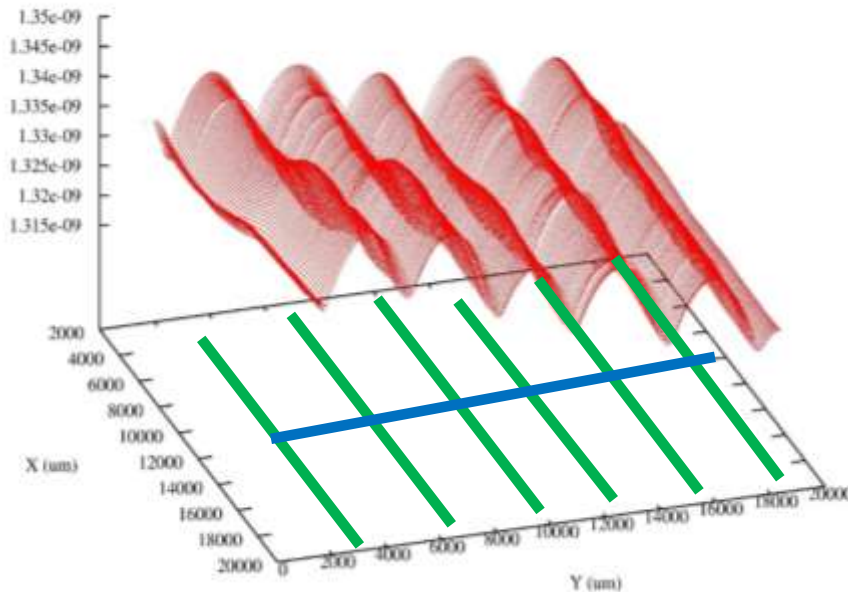- Saves significant runtime since extraction alone can take a day or more

RESULTS

The tuning flow allowed us to reduce the structural skew of the mesh

- **Skew was reduced to < 30ps across the whole core region and across multiple process corners**
  (from > 100ps before tuning)

The removal of the majority of the cross-links also helped save power

- **Power consumed by the distribution (including buffers in the vertical spine + horizontal ribs) was = 1.4W for a 16mm X17mm clock mesh area at 0.9V, 800Mhz**
- **Removal of the horizontal cross-links helped reduce mesh capacitance and thus clock power by 30%**

Skew   = 30ps
Core clock area   = 16mm X 17mm

* Example of skew plot over
  an 16mm X 17mm core region

CONCLUSION

We have presented a 2 step tuning flow that can de-skew and tune a clock mesh containing several thousand buffers

- The fine-tuning flow enables 2 to 3 iterations to be completed within 24 hours.
- Structural skew of more than 100ps was reduced to less than 25ps

Removal of horizontal Mz cross-links in the clock mesh helped reduce clock power

- Clock distribution + mesh consumed a total of 1.4W in a 100W chip
- The removal of most of the horizontal cross-links reduced mesh capacitance and power by ~30%

This tuning flow was used in multiple chips across two technology generations

Thank You