Configurable BISR Chain for Fast Repair Data Loading

Wei Zou **Benoit Nadeau-Dostie**

Siemens Digital Industries Software

SIEMENS

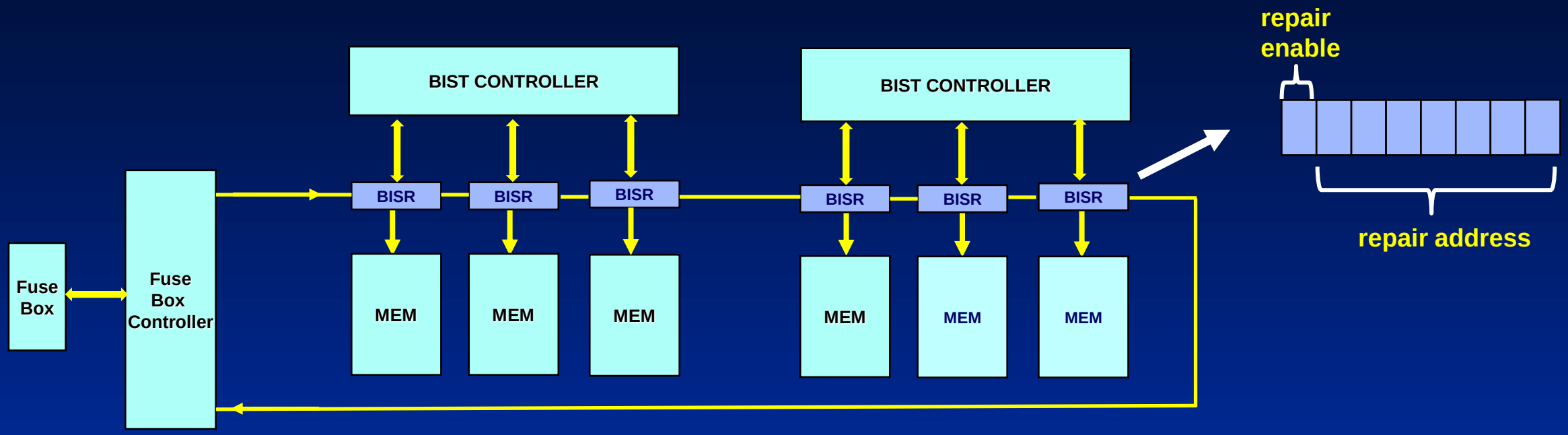
Motivation

- Today’s designs may have tens of thousands of memories with repair redundancy
- It takes a long time to load the repair data serially during system power-up
- Can we take advantage of the fact that very few of the memories actually need repair to significantly speed up the process?

Outline

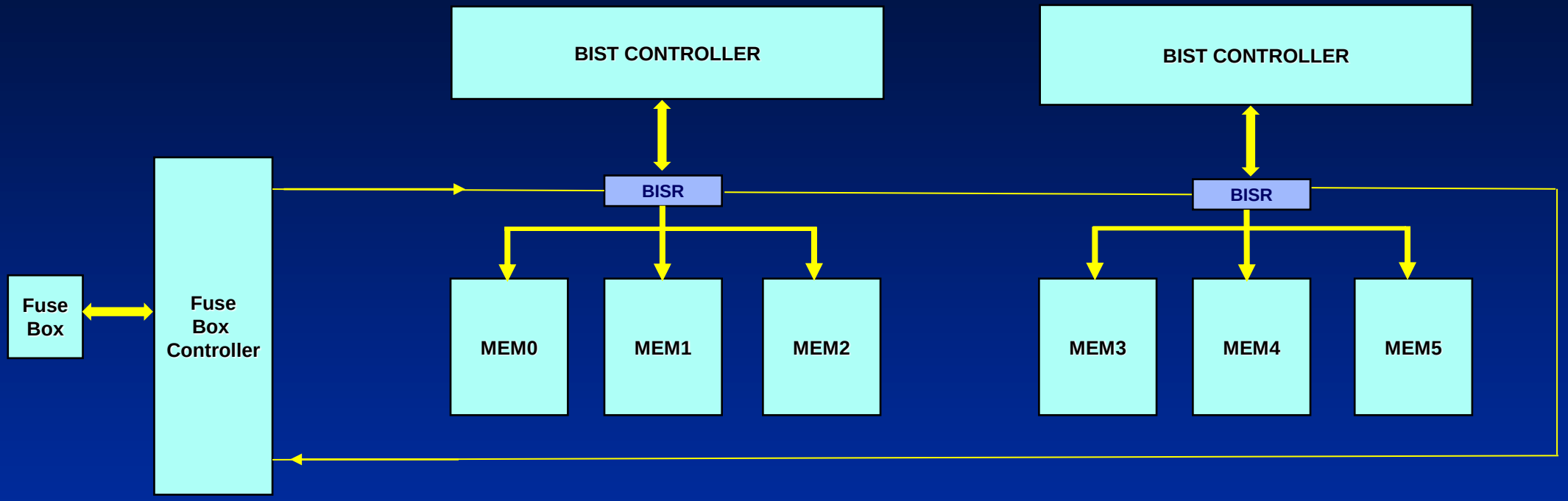
- The general memory repair system
- Prior work
- Configurable BISR chain repair system
- Experimental results
- Conclusions

The general memory repair system



- Dedicated repair register for each memory
- Repair enable indicates that memory needs repair
- Most repair registers contain only contain 0s and allow compression of repair information in fuse box

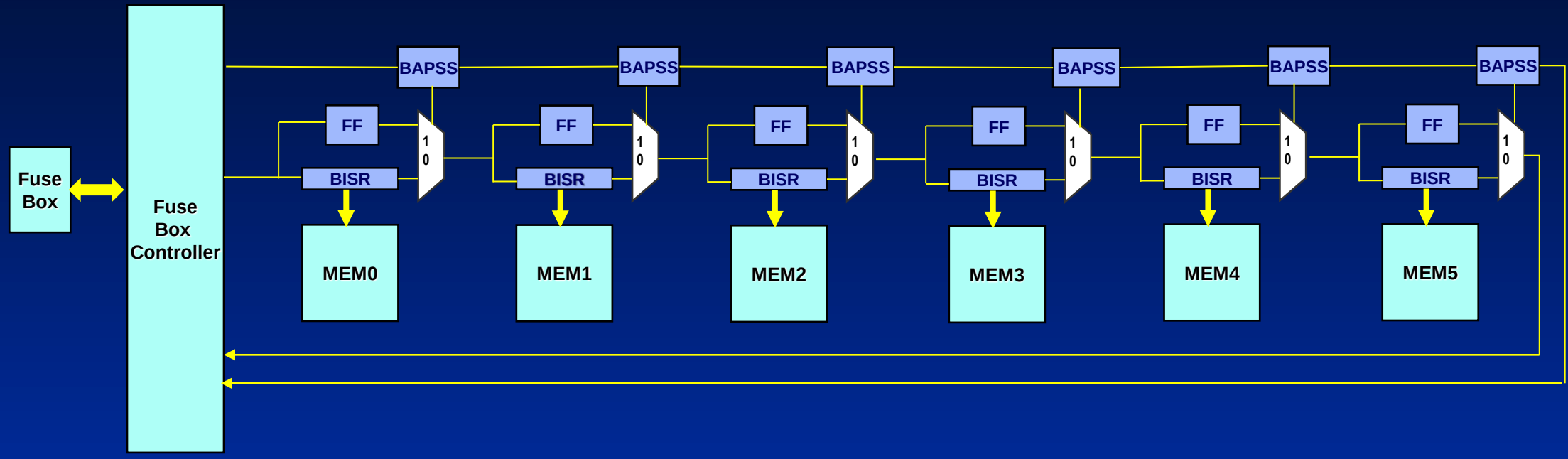
Prior work (repair sharing)



- Use same repair solution for several memories
- Good results obtained for memories using row repair and memories behind a shared bus
- Limited application for distributed memories using column repair
- Potential yield loss

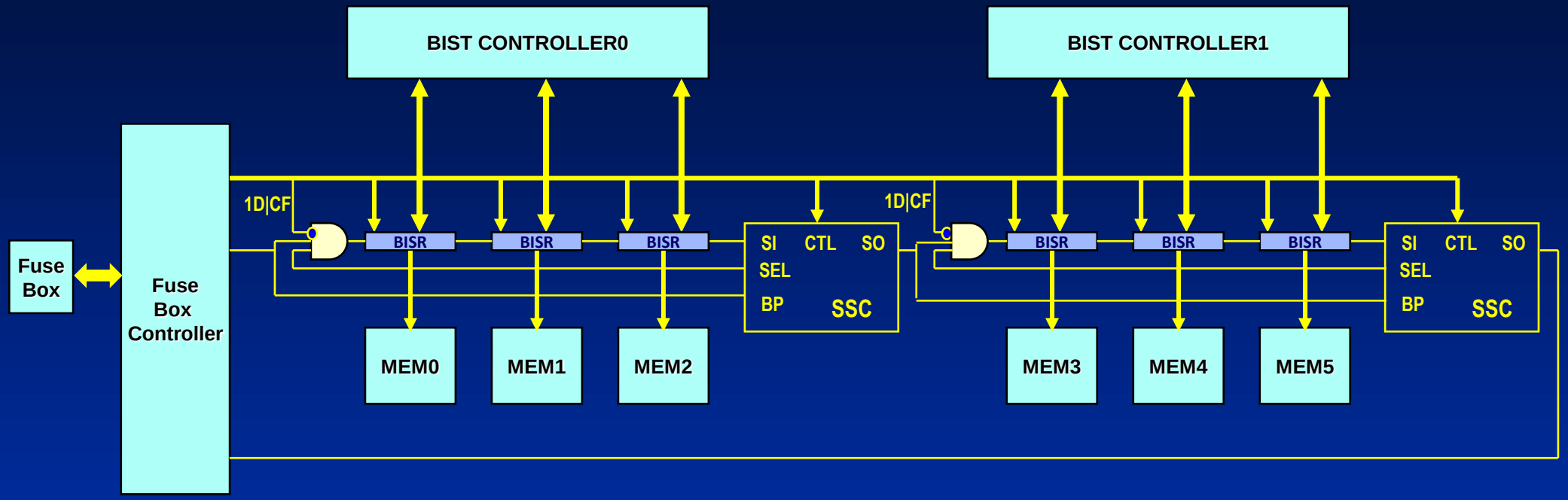
Prior work (memory bypass)

Devanathan
DAC 2013



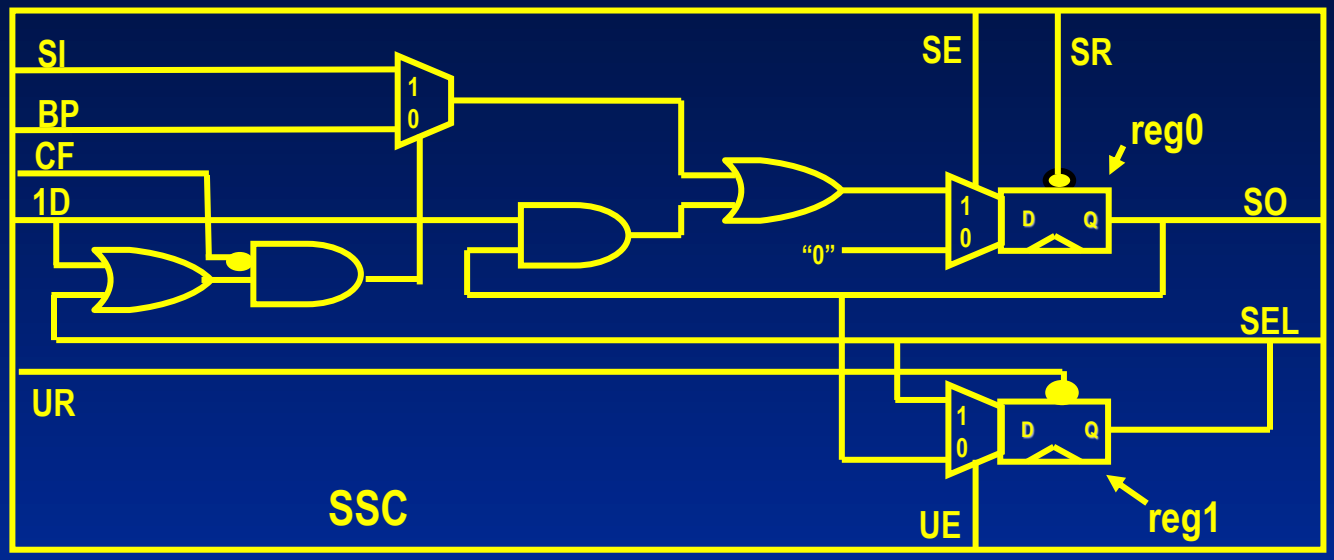
- Each repair register can be bypassed
- Configuration chain loaded first to select repair registers to include in chain
- Pipeline flop needed to avoid long asynchronous paths
- Speedup limited to about 5X

Configurable BISR chain repair system



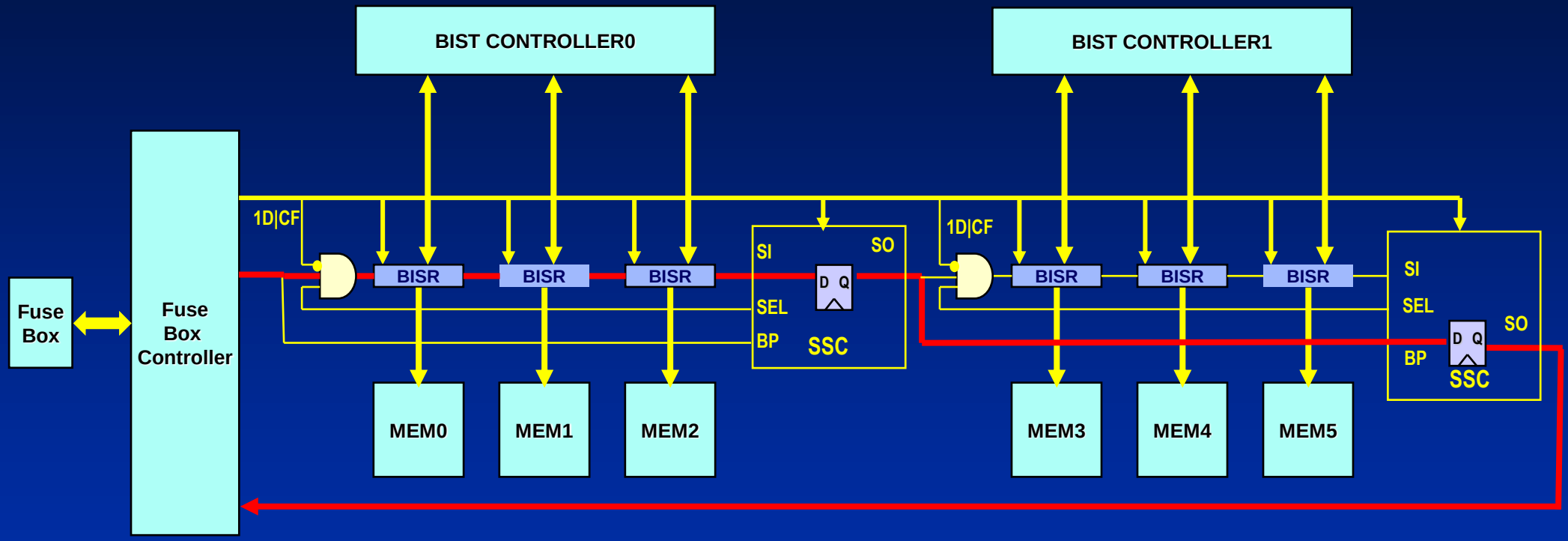
- Extend Devanathan's idea to bypass several repair registers at a time
- Bypassing longer segments reduces the overhead associated to the configuration chain

Segment selection circuit (SSC)



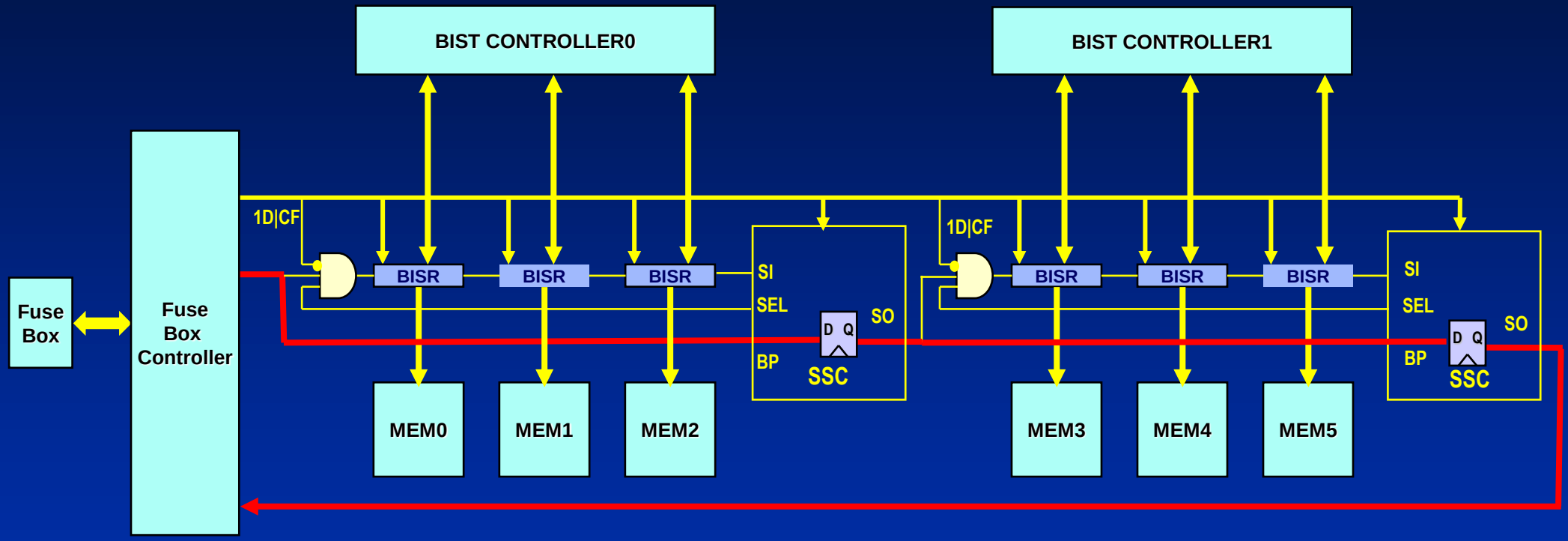
- Structure similar to Segment Insertion Bit (SIB) of IEEE 1687
  - Selects/bypasses associated chain segment
- SSC has additional circuitry to identify segments that need repair
  - 1-detection logic

Active scan path with bypassed segment



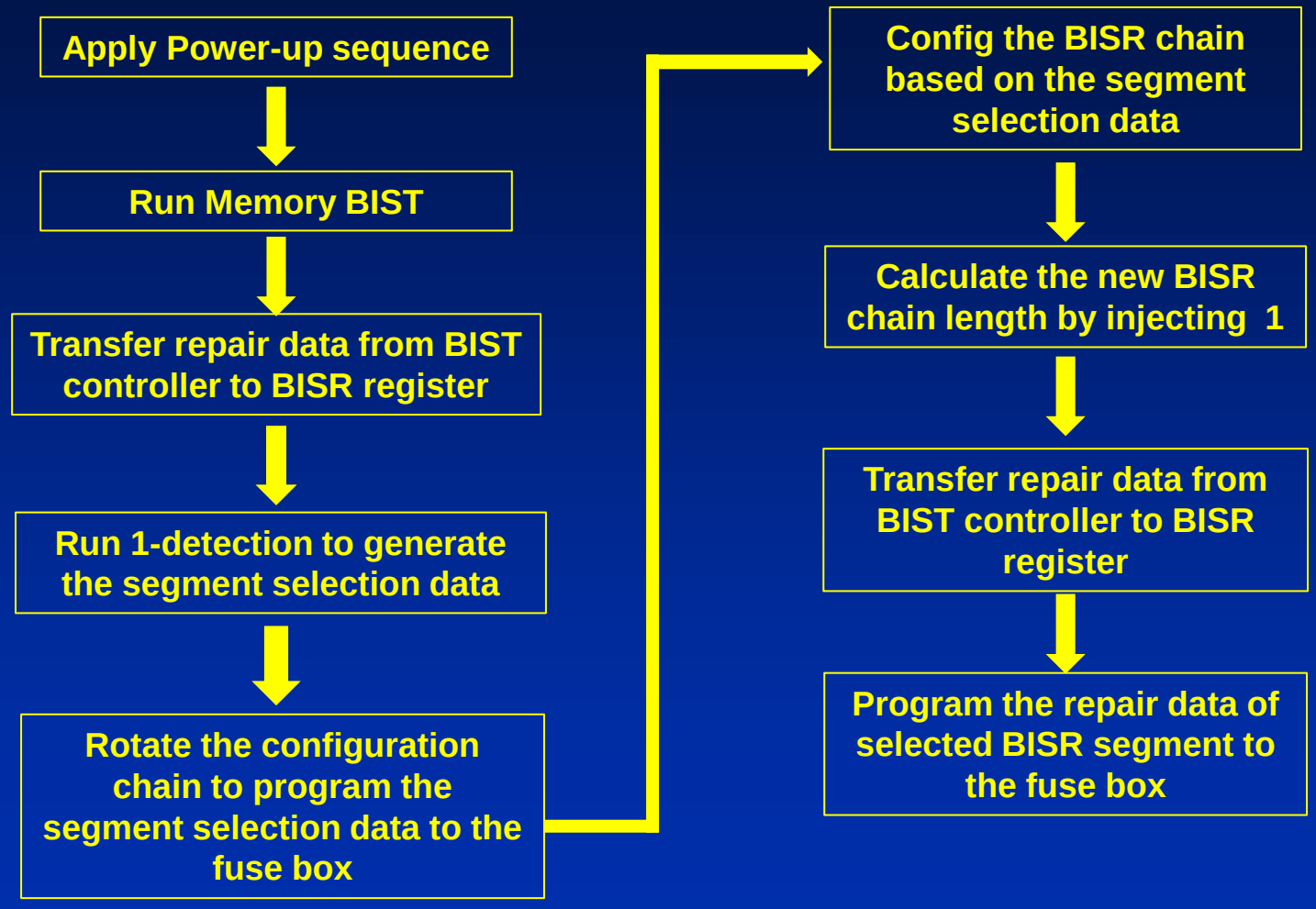
- Left segment included in scan path because at least one memory needs repair
- Right segment bypassed because none of the memories need repair
  - Segment input forced to 0

Active scan path (configuration chain selected)

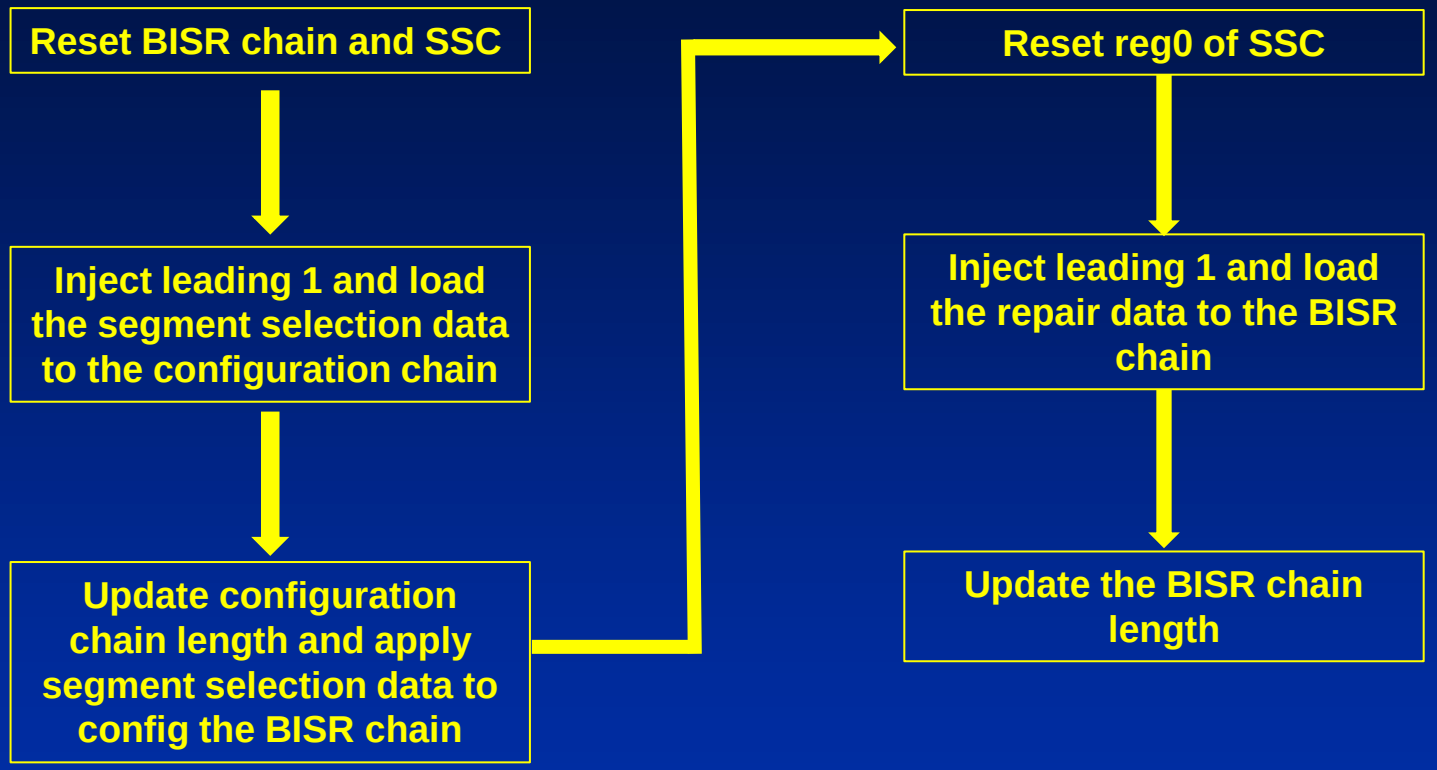


- All segments are bypassed to load chain configuration

Repair data programming sequence



Power-up sequence



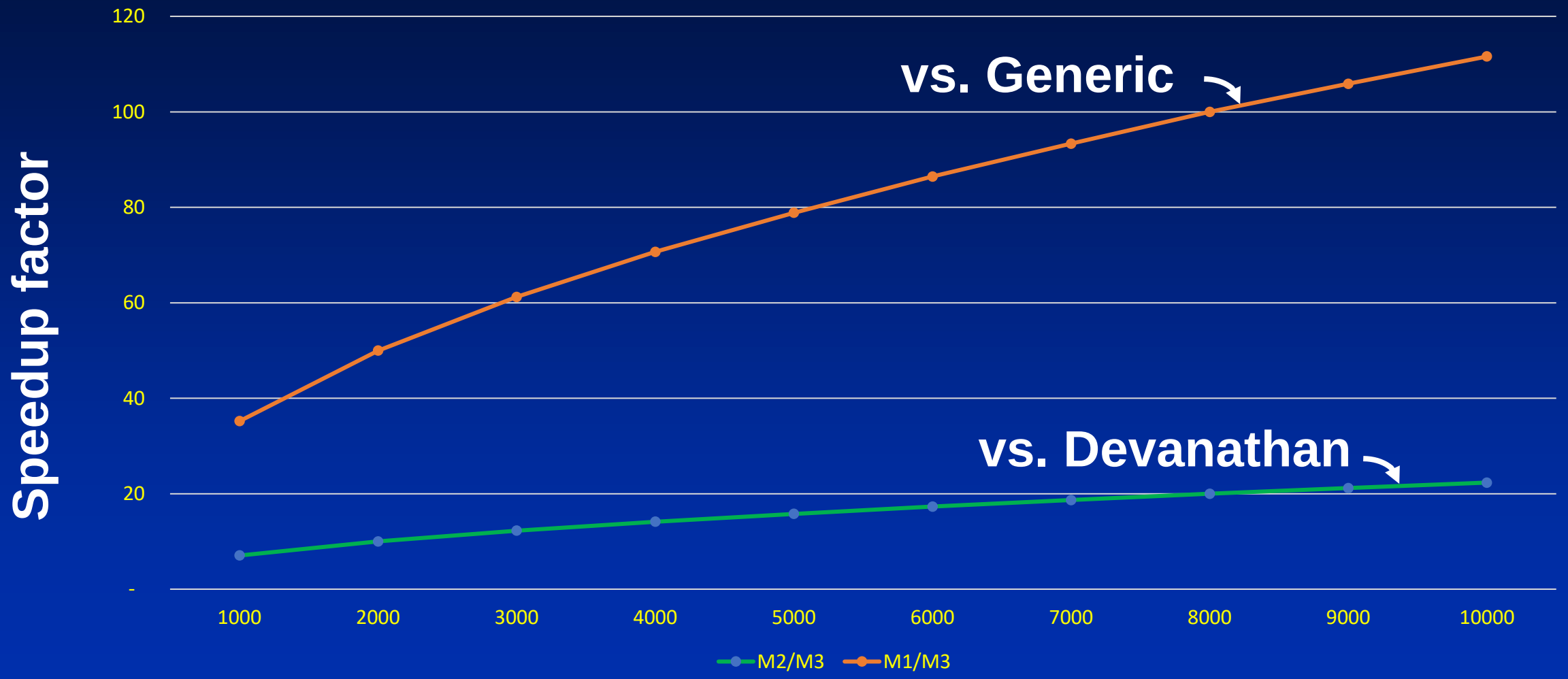
Partitioning algorithm considerations

- Number of segments depends on a few factors
- Most important one is defect density
  - High defect density requires shorter segments to reduce probability of having to include a segment
- Segments of pre-existing IP blocks must be included as is
  - Not always possible to implement optimal segment size

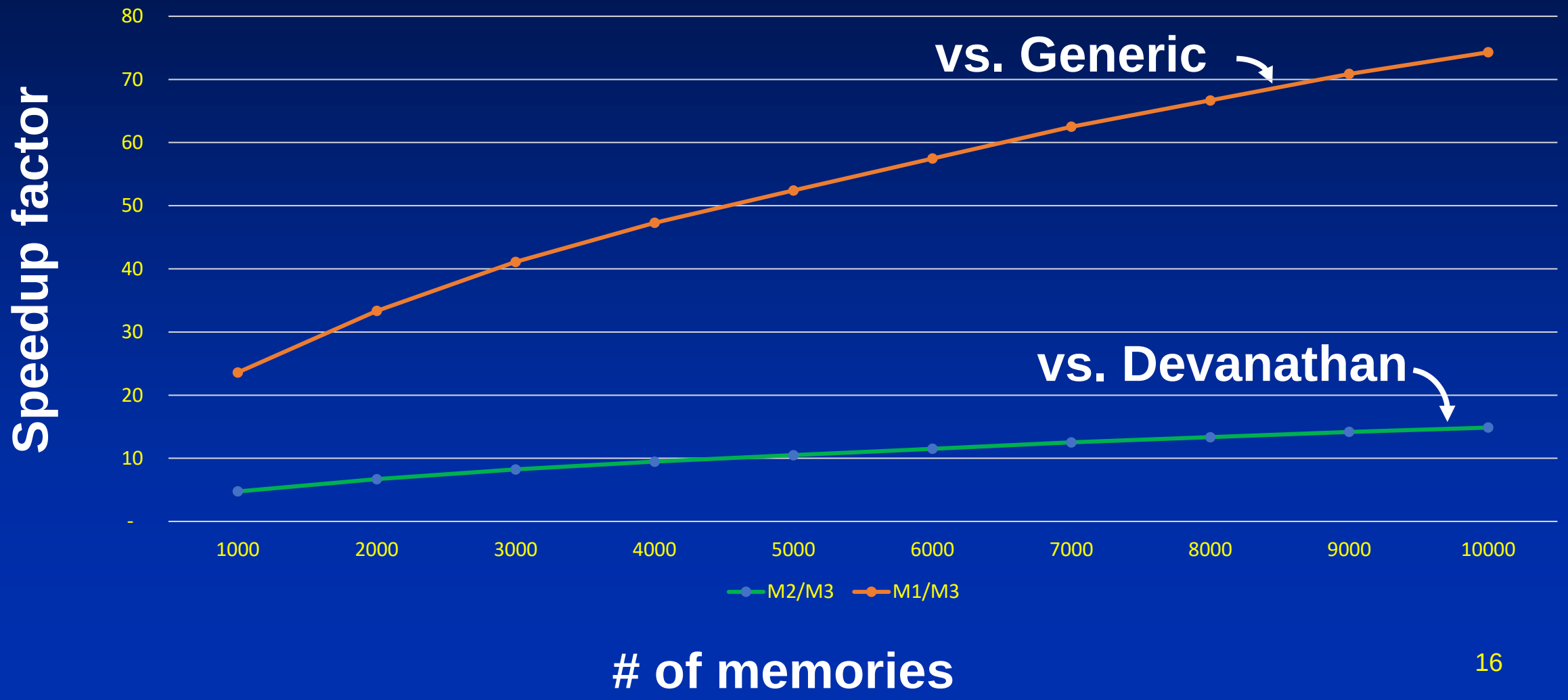
Calculation of optimal segment size

- The BISR Chain Shifting time *T = Nrepair X L/Nseg + 2 X Nseg*
  - L: the total length of the repair registers
  - *Nseg:* number of segments
  - *Nrepair* : number of segments requiring repair
- To minimize T
  - *(Nrepair X L / Nseg + 2 X Nseg)' = 0*
- $Optimal\ number\ of\ segments\colon\ Nseg = \sqrt{Nrepair\ X\ L/2}$
- $Optimal\ Segment\ size\ Nsize = L/Nseg$

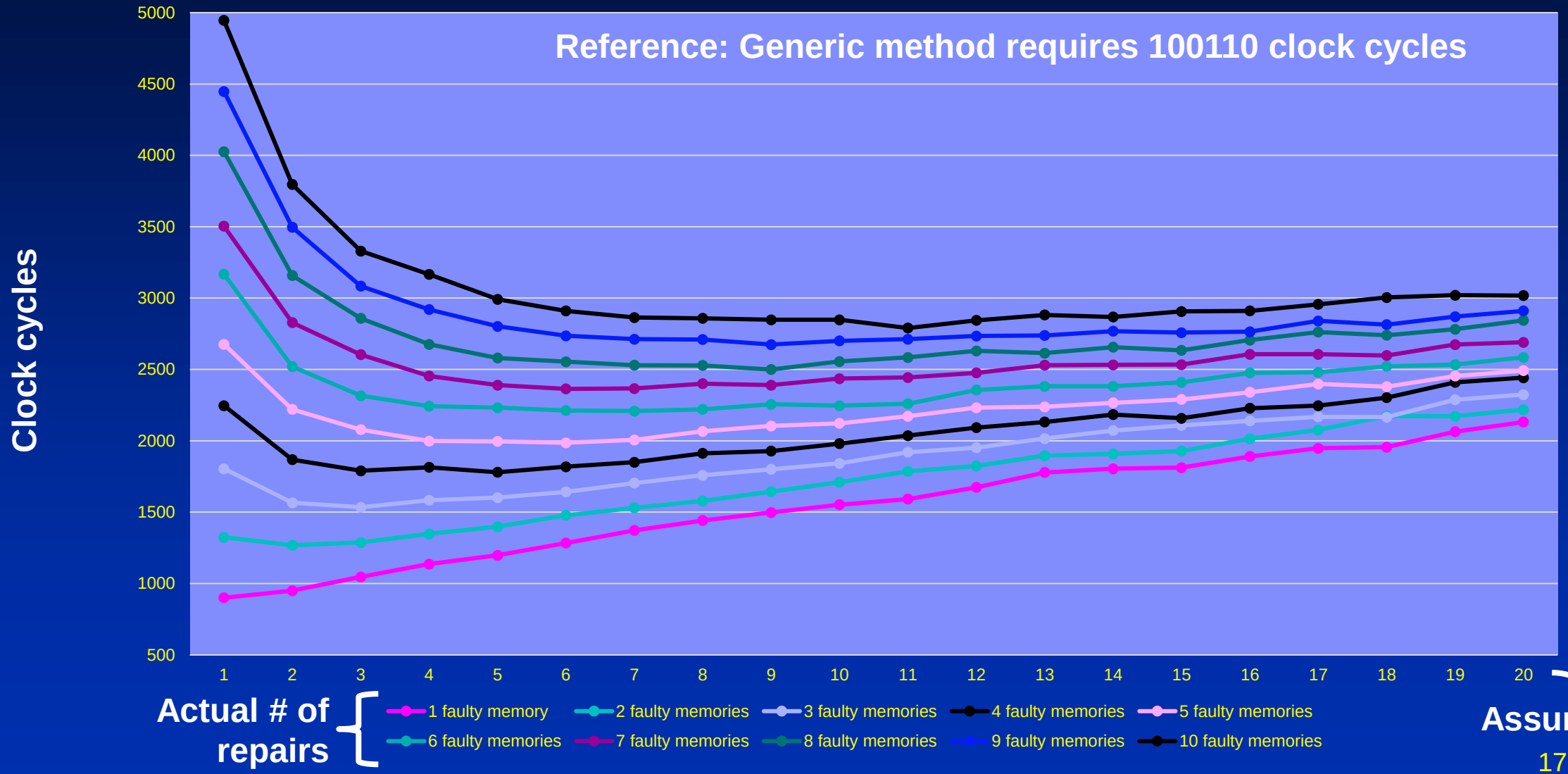
Repair data loading speedup factor (single repair)



Repair data loading speedup factor (two repairs)



Repair data loading cycles
(assumed vs actual number of repairs)



Conclusions

- A configurable BISR chain repair system is proposed to speed up repair data loading during chip power-up
- Experimental results show that number of clock cycles can be reduced by one to two orders of magnitude compared to previous methods