VPR Assessment of a Novel Partitioning Algorithm

David Munro

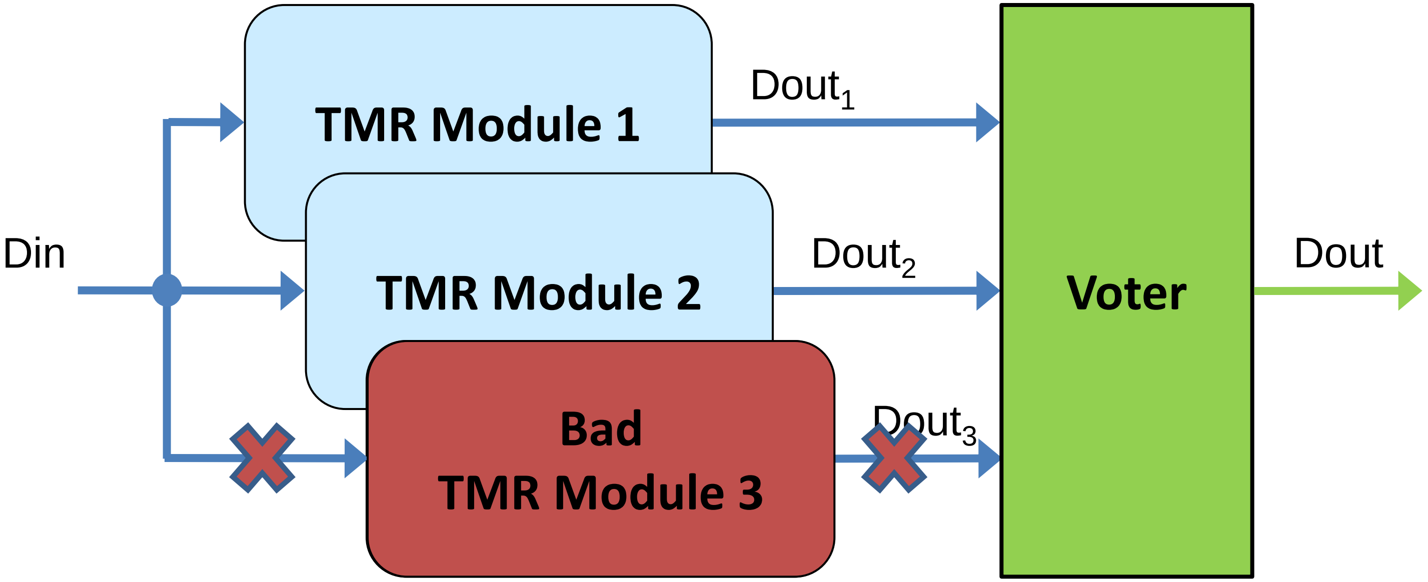
Supervised by Oliver Diessel

Overview

- Why and What? Quick discussion of motivation, and what this thesis is actually doing.
- How? Very high level look at the implementation.
- Results. The interesting bit.
- Limitations. Every project has them.

Why?

- FPGAs are awesome…but vulnerable to configuration errors caused by e.g. space radiation, that ASICs aren't.
- TMR with partial reconfiguration provides redundancy and error correction.
- Make three copies of a circuit, and feed the outputs to a voter.
- Once an error is detected we can fix it by selectively reconfiguring the incorrect module.
- Need to detect, reconfigure and resynchronise within error rate.



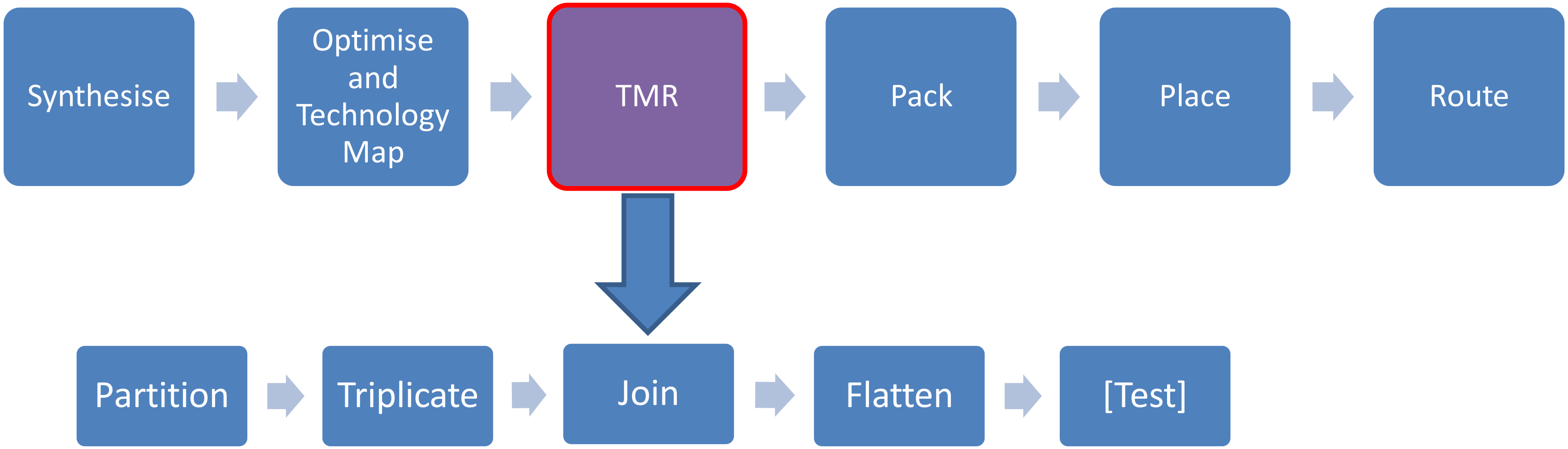
Source Cetin & Diessel (2012)

Why

- TMR entire circuit -> lower overhead, lower fault tolerance (higher recovery time).
- TMR every component in circuit -> higher fault tolerance, higher overhead (4x instead of 3x).
- Middle ground, TMR sections such that we meet target fault tolerance.
- What is a reasonable Error Recovery Time? User specified, depends on application. For space based applications 1.4s mean time between errors[1].

[1] P. J. Pingree, "Advancing NASA's on-board processing capabilities with reconfigurable fpga technologies". Aerospace Technologies Advancements.

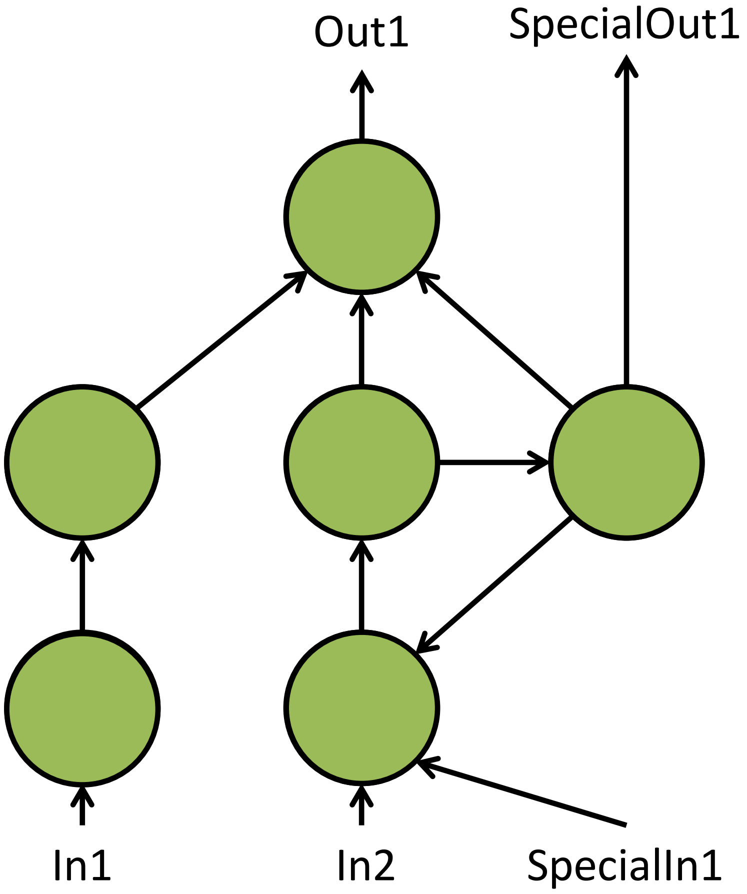
Toolchain



- Insert new ‘TMR’ step in normal CAD toolchain.
- TMR consists of several substeps, each a separate module.
- Triplicate and Join treat input as black box, create ‘glue’ from template to wire appropriate bits together in a hierarchical netlist.
- Partition is more complicated.
- Flatten turns netlist back into format for VPR.
- Test verifies generated circuit is equivalent to the original.

How? - Partition

- Start at output, traverse circuit depth first.
- Visit each element once and only once. As you do, insert it into current partition.
- If inserting the element creates a cycle, cut it.
- If it would exceed recovery time, write the partition out to file, create a new empty partition, and insert into that one instead.
- Repeat until entire circuit is partitioned.



Recovery Time

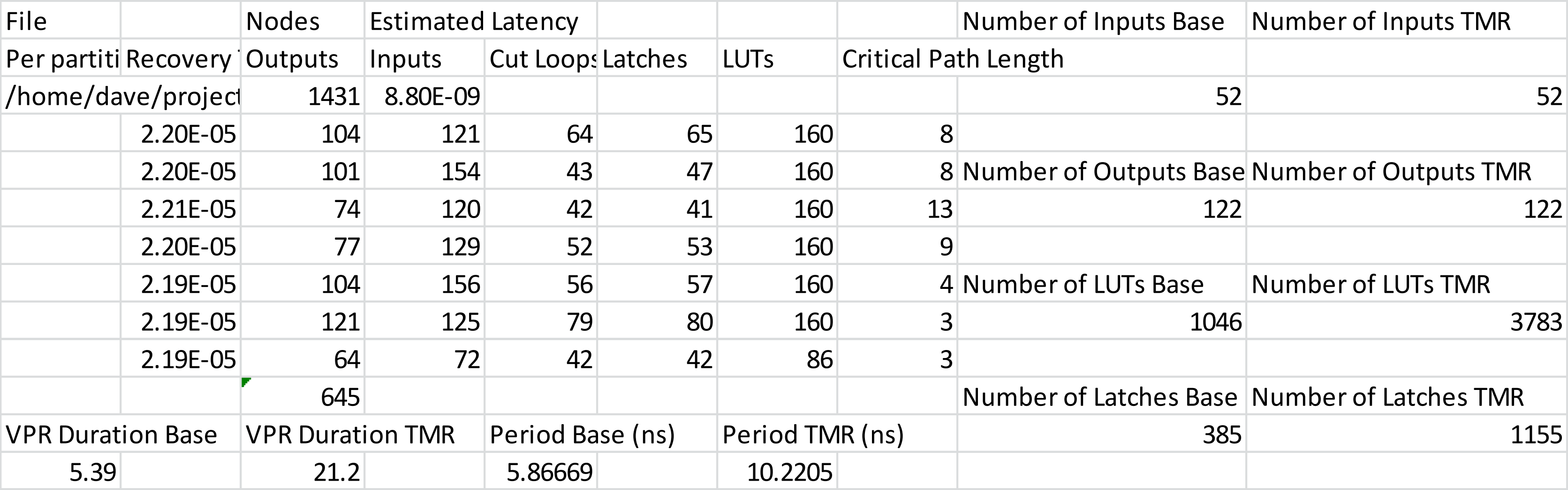
- Recovery time is error detection time + communication delay + reconfiguration time + resynchronisation time.

$$\begin{aligned}
&\mathit{ErrorDetectionTime} \leq \mathit{Latency} = \mathit{ClockPeriod} \times \mathit{PipelineSteps} \approx 10^{-7}\mathrm{s}\\
&\mathit{CommunicationsDelay} \leq 50 \times 2(\mathit{NumPartitions} + 1) \times \mathit{ClockPeriod}\\
&= 100 \times \mathit{ClockPeriod} \times (\mathit{NumPartitions} + 1) \approx 10^{-5}\\
&\mathit{ReconfigurationTime} = \left\lceil \frac{\max(\mathit{LUTs}, \mathit{Latches})}{160} \right\rceil \times 1.48 \times 10^{-5} \approx 10^{-4}\\
&\mathit{ResynchronisationTime} \leq \mathit{Latency} = \mathit{ClockPeriod} \times \mathit{PipelineSteps} \approx 10^{-7}\mathrm{s}
\end{aligned}$$

- Reconfiguration Time dominates

Results

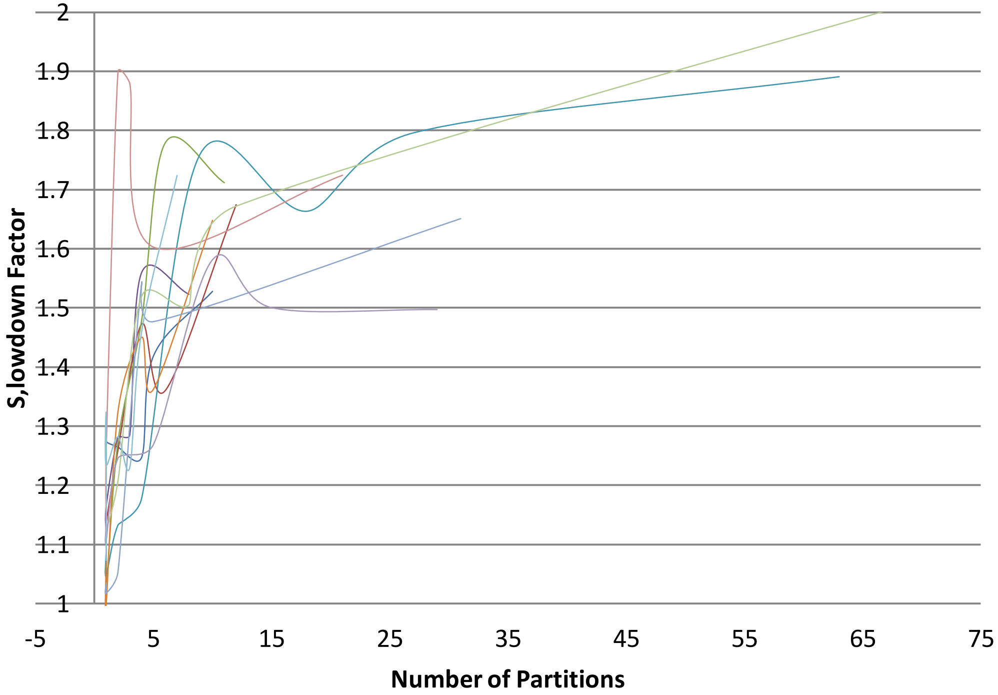
- Some results use old formula for RecoveryTime -> Number of partitions and RecoveryTime may be different in final results. Overall trends and results still hold. No major changes.
- Area usage is 3x + a bit. Unsurprising, but a nice sanity check.
- SUM(Outputs)=645. NumLUTsTMR =3*Base+645
- LatchesTMR=3xBase

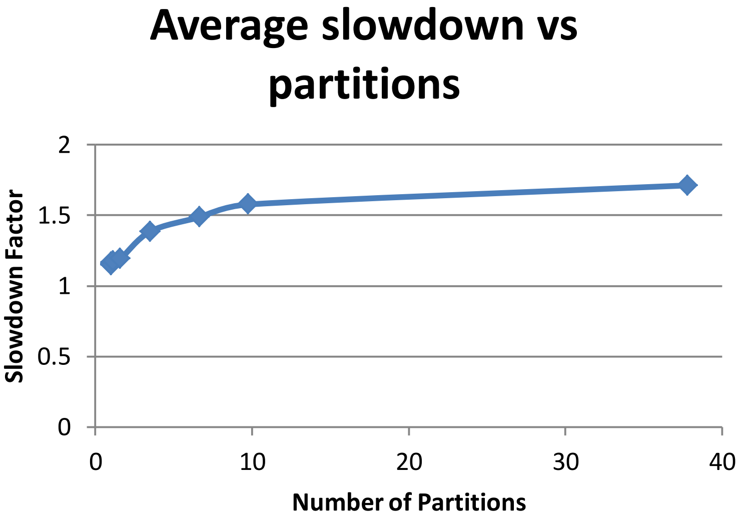


File		Nodes	Estimated Latency					Number of Inputs Base	Number of Inputs TMR
Per partiti	Recovery '	Outputs	Inputs	Cut Loops	Latches	LUTs	Critical Path Length		
/home/dave/project		1431	8.80E-09					52	52
	2.20E-05	104	121	64	65	160	8		
	2.20E-05	101	154	43	47	160	8	Number of Outputs Base	Number of Outputs TMR
	2.21E-05	74	120	42	41	160	13	122	122
	2.20E-05	77	129	52	53	160	9		
	2.19E-05	104	156	56	57	160	4	Number of LUTs Base	Number of LUTs TMR
	2.19E-05	121	125	79	80	160	3	1046	3783
	2.19E-05	64	72	42	42	86	3		
		645						Number of Latches Base	Number of Latches TMR
VPR Duration Base		VPR Duration TMR		Period Base (ns)		Period TMR (ns)		385	1155
5.39		21.2		5.86669		10.2205			

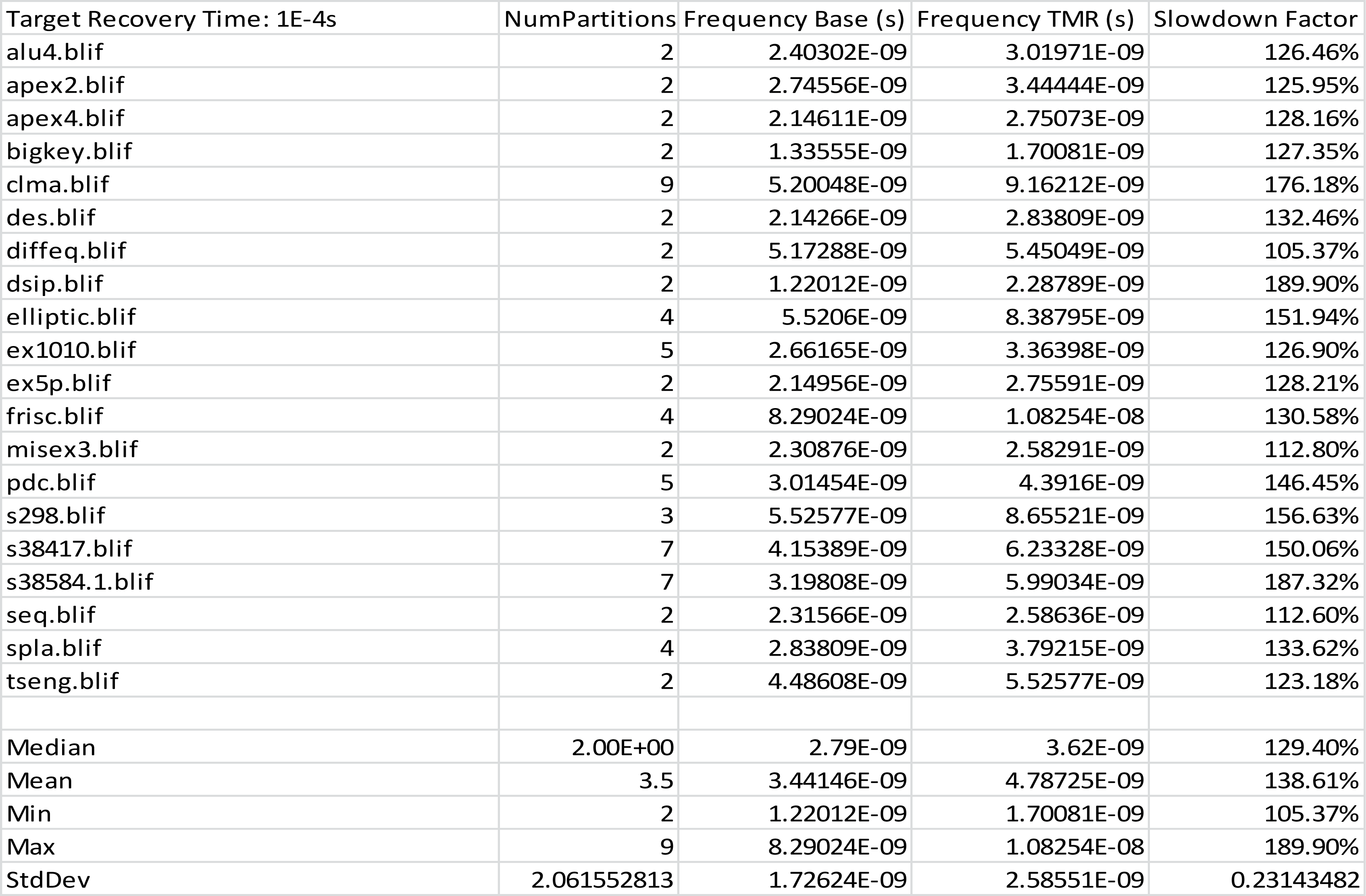
Results

- Latency is interesting i.e. hard to predict. Very circuit dependent.
- In general, the more partitions, the slower.





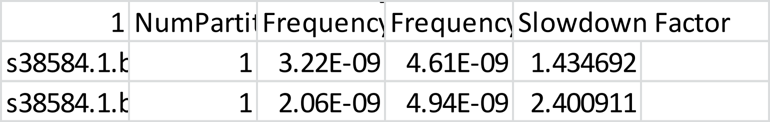
Results



Target Recovery Time: 1E-4s	NumPartitions	Frequency Base (s)	Frequency TMR (s)	Slowdown Factor
alu4.blif	2	2.40302E-09	3.01971E-09	126.46%
apex2.blif	2	2.74556E-09	3.44444E-09	125.95%
apex4.blif	2	2.14611E-09	2.75073E-09	128.16%
bigkey.blif	2	1.33555E-09	1.70081E-09	127.35%
clma.blif	9	5.20048E-09	9.16212E-09	176.18%
des.blif	2	2.14266E-09	2.83809E-09	132.46%
diffeq.blif	2	5.17288E-09	5.45049E-09	105.37%
dsip.blif	2	1.22012E-09	2.28789E-09	189.90%
elliptic.blif	4	5.5206E-09	8.38795E-09	151.94%
ex1010.blif	5	2.66165E-09	3.36398E-09	126.90%
ex5p.blif	2	2.14956E-09	2.75591E-09	128.21%
frisc.blif	4	8.29024E-09	1.08254E-08	130.58%
misex3.blif	2	2.30876E-09	2.58291E-09	112.80%
pdc.blif	5	3.01454E-09	4.3916E-09	146.45%
s298.blif	3	5.52577E-09	8.65521E-09	156.63%
s38417.blif	7	4.15389E-09	6.23328E-09	150.06%
s38584.1.blif	7	3.19808E-09	5.99034E-09	187.32%
seq.blif	2	2.31566E-09	2.58636E-09	112.60%
spla.blif	4	2.83809E-09	3.79215E-09	133.62%
tseng.blif	2	4.48608E-09	5.52577E-09	123.18%
Median	2.00E+00	2.79E-09	3.62E-09	129.40%
Mean	3.5	3.44146E-09	4.78725E-09	138.61%
Min	2	1.22012E-09	1.70081E-09	105.37%
Max	9	8.29024E-09	1.08254E-08	189.90%
StdDev	2.061552813	1.72624E-09	2.58551E-09	0.23143482

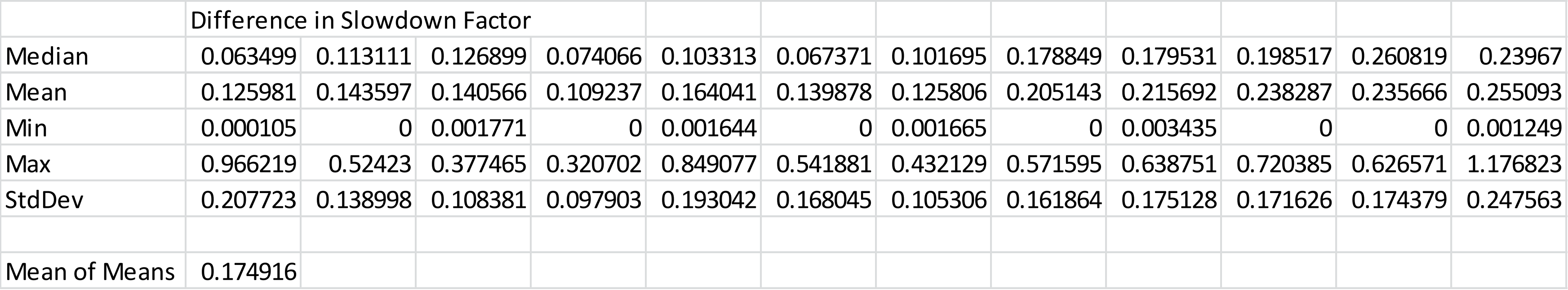
Results

- Placer uses simulated annealing -> Random factor.
- Generally runs are within 10-20%, but some outliers.



1	NumPartit	Frequency	Frequency	Slowdown Factor	
s38584.1.b	1	3.22E-09	4.61E-09	1.434692	
s38584.1.b	1	2.06E-09	4.94E-09	2.400911	

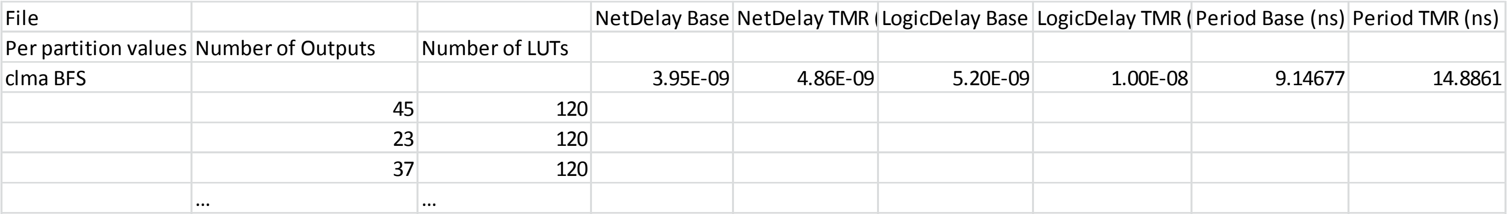
- Need to take average across multiple runs (aiming for 10 runs).



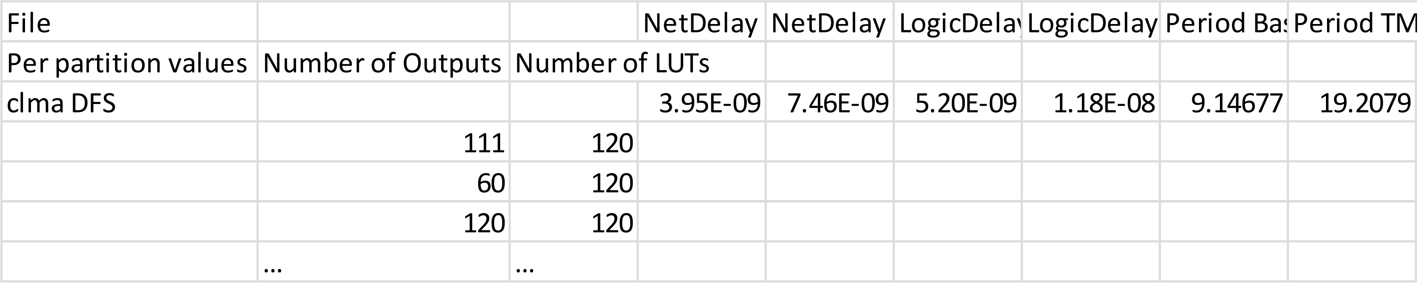
	Difference in Slowdown Factor											
Median	0.063499	0.113111	0.126899	0.074066	0.103313	0.067371	0.101695	0.178849	0.179531	0.198517	0.260819	0.23967
Mean	0.125981	0.143597	0.140566	0.109237	0.164041	0.139878	0.125806	0.205143	0.215692	0.238287	0.235666	0.255093
Min	0.000105	0	0.001771	0	0.001644	0	0.001665	0	0.003435	0	0	0.001249
Max	0.966219	0.52423	0.377465	0.320702	0.849077	0.541881	0.432129	0.571595	0.638751	0.720385	0.626571	1.176823
StdDev	0.207723	0.138998	0.108381	0.097903	0.193042	0.168045	0.105306	0.161864	0.175128	0.171626	0.174379	0.247563
Mean of Means	0.174916											

Results

- We traverse circuit to add nodes depth first i.e. explore children before finishing current level.
- Significantly better results than breadth first. Far fewer voters and wires between partitions, so less slowdown.



File			NetDelay Base	NetDelay TMR (	LogicDelay Base	LogicDelay TMR (	Period Base (ns)	Period TMR (ns)
Per partition values	Number of Outputs	Number of LUTs						
clma BFS			3.95E-09	4.86E-09	5.20E-09	1.00E-08	9.14677	14.8861
	45	120						
	23	120						
	37	120						
	…	…						



File			NetDelay	NetDelay	LogicDela	LogicDelay	Period Ba	Period TM
Per partition values	Number of Outputs	Number of LUTs						
clma DFS			3.95E-09	7.46E-09	5.20E-09	1.18E-08	9.14677	19.2079
	111	120						
	60	120						
	120	120						
	…	…						

Limitations

- Implementation to evaluate effectiveness, missing some functionality.
- E.g. Can’t constrain placement to ensure partitions are actually mapped to separate reconfiguration areas.
- Our target architecture is still a theoretical simplified version. Only latches and LUTs.
- Partitioner doesn’t try very hard to be optimal.