Embench™: An Evolving Benchmark Suite for Embedded IoT Computers from an Academic-Industrial Cooperative

Recruiting for the Long Overdue and Deserved Demise of Dhrystone

(+ Bonus Announcement at End of Talk)

Jeremy Bennett (Embecosm), Palmer Dabbelt (SiFive),
Cesare Garlati (Hex Five Security),
G. S. Madhusudan (India Institute of Technology Madras),
Trevor Mudge (University of Michigan), and
David Patterson (University of California Berkeley)

June 12, 2019

State of Benchmarks for IoT/Embedded Computers

- Billions of Internet of Things (IoT) devices shipped soon
- Still no high quality, widely reported benchmark for embedded computers

The image part with relationship ID rId5 was not found in the file.

Yunsup Lee, SiFive CTO, Keynote address "Opportunities and Challenges of Building Silicon in the Cloud" 12/5/18 RISC-V Summit:

"... *the benchmark scores are 4.9 CoreMarks/MHz and 2.5 DMIPS/MHz. I'm saying this in front of Dave [Patterson], who doesn't really like Dhrystone or CoreMark as benchmarks. Sorry. This is the industry standard benchmark I learned.*"

It's past time to apologize; let's fix!

Learning from Benchmark History

- Incorporate good ideas and avoid mistakes of past benchmarks
- First learn from features of widely quoted benchmarks:

Learning from Benchmark History

- Incorporate good ideas and avoid mistakes of past benchmarks
- First learn from features of widely quoted benchmarks:

	Linpack	*Dhrystone*	*SPEC CPU*	*CoreMark*	*MLPerf 0.5*
Year	1977	1984	1989	2009	2018
Initial Target	Supercomputer	Systems programming	Unix Server	Embedded	Server (ML training)
Type	Library	Synthetic	Applications	Synthetic	Applications

Learning from Benchmark History

- Incorporate good ideas and avoid mistakes of past benchmarks
- First learn from features of widely quoted benchmarks:

	Linpack	*Dhrystone*	*SPEC CPU*	*CoreMark*	*MLPerf 0.5*
Year	1977	1984	1989	2009	2018
Initial Target	Supercomputer	Systems programming	Unix Server	Embedded	Server (ML training)
Type	Library	Synthetic	Applications	Synthetic	Applications
Quality Reputation	Low	Low	**High**	Low	**High**

Learning from Benchmark History

- Incorporate good ideas and avoid mistakes of past benchmarks
- First learn from features of widely quoted benchmarks:

	Linpack	*Dhrystone*	*SPEC CPU*	*CoreMark*	*MLPerf 0.5*
Year	1977	1984	1989	2009	2018
Initial Target	Supercomputer	Systems programming	Unix Server	Embedded	Server (ML training)
Type	Library	Synthetic	Applications	Synthetic	Applications
Quality Reputation	Low	Low	**High**	Low	**High**
Free (no cost)	✓	✓	$1000 ($250 academia)	✓	✓

Learning from Benchmark History

- Incorporate good ideas and avoid mistakes of past benchmarks
- First learn from features of widely quoted benchmarks:

	Linpack	*Dhrystone*	*SPEC CPU*	*CoreMark*	*MLPerf 0.5*
Year	1977	1984	1989	2009	2018
Initial Target	Supercomputer	Systems programming	Unix Server	Embedded	Server (ML training)
Type	Library	Synthetic	Applications	Synthetic	Applications
Quality Reputation	Low	Low	**High**	Low	**High**
Free (no cost)	✓	✓	$1000 ($250 academia)	✓	✓
Easy to Port	✓	✓	✓	✓	

Learning from Benchmark History

- Incorporate good ideas and avoid mistakes of past benchmarks
- First learn from features of widely quoted benchmarks:

	Linpack	*Dhrystone*	*SPEC CPU*	*CoreMark*	*MLPerf 0.5*
Year	1977	1984	1989	2009	2018
Initial Target	Supercomputer	Systems programming	Unix Server	Embedded	Server (ML training)
Type	Library	Synthetic	Applications	Synthetic	Applications
Quality Reputation	Low	Low	**High**	Low	**High**
Free (no cost)	✓	✓	$1000 ($250 academia)	✓	✓
Easy to Port	✓	✓	✓	✓	
Revision Frequency	None since 1991	None since 1988	**3 years**	Never	**1 year (planned)**

Learning from Benchmark History

- Incorporate good ideas and avoid mistakes of past benchmarks
- First learn from features of widely quoted benchmarks:

	Linpack	*Dhrystone*	*SPEC CPU*	*CoreMark*	*MLPerf 0.5*
Year	1977	1984	1989	2009	2018
Initial Target	Supercomputer	Systems programming	Unix Server	Embedded	Server (ML training)
Type	Library	Synthetic	Applications	Synthetic	Applications
Quality Reputation	Low	Low	**High**	Low	**High**
Free (no cost)	✓	✓	$1000 ($250 academia)	✓	✓
Easy to Port	✓	✓	✓	✓	
Revision Frequency	None since 1991	None since 1988	**3 years**	Never	**1 year (planned)**
Number of Programs	1	1	**10 SPEC89 - 23 SPEC2017**	1	**7**

SPEC Benchmark History: SPEC89 - SPEC2017

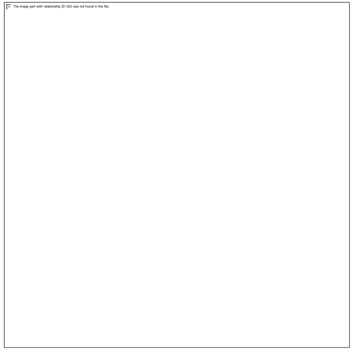
The image part with relationship ID rId3 was not found in the file.

All 82 programs from 6 SPEC generations

Only 3 integer programs and 3 FP programs survived ≥ 3 generations

Fig 1.17, *Computer Architecture: A Quantitative Approach*, 6th Edition, 2018

Learning from Benchmark History



- Incorporate good ideas and avoid mistakes of past benchmarks
- First learn from features of widely quoted benchmarks:

	Linpack	*Dhrystone*	*SPEC CPU*	*CoreMark*	*MLPerf 0.5*
Year	1977	1984	1989	2009	2018
Initial Target	Supercomputer	Systems programming	Unix Server	Embedded	Server (ML training)
Type	Library	Synthetic	Applications	Synthetic	Applications
Quality Reputation	Low	Low	**High**	Low	**High**
Free (no cost)	✓	✓	$1000 ($250 academia)	✓	✓
Easy to Port	✓	✓	✓	✓	
Revision Frequency	None since 1991	None since 1988	**3 years**	Never	**1 year (planned)**
Number of Programs	1	1	**10 SPEC89 - 23 SPEC2017**	1	**7**
Organization to Evolve Benchmark	✓		✓	✓	✓

Learning from Benchmark History

- Incorporate good ideas and avoid mistakes of past benchmarks
- First learn from features of widely quoted benchmarks:

	Linpack	*Dhrystone*	*SPEC CPU*	*CoreMark*	*MLPerf 0.5*
Year	1977	1984	1989	2009	2018
Initial Target	Supercomputer	Systems programming	Unix Server	Embedded	Server (ML training)
Type	Library	Synthetic	Applications	Synthetic	Applications
Quality Reputation	Low	Low	**High**	Low	**High**
Free (no cost)	✓	✓	$1000 ($250 academia)	✓	✓
Easy to Port	✓	✓	✓	✓	
Revision Frequency	None since 1991	None since 1988	**3 years**	Never	**1 year (planned)**
Number of Programs	1	1	**10 SPEC89 - 23 SPEC2017**	1	**7**
Organization to Evolve Benchmark	✓		✓	✓	✓
Single Summary Score	FLOPS/second	Speed Ratio	**Geometric Mean Speed Ratios**	Speed Ratio	Weighted Mean Ratios **+ Std. Dev.**

Learning from Benchmark History

- Incorporate good ideas and avoid mistakes of past benchmarks
- First learn from features of widely quoted benchmarks:

	Linpack	*Dhrystone*	*SPEC CPU*	*CoreMark*	*MLPerf 0.5*
Year	1977	1984	1989	2009	2018
Initial Target	Supercomputer	Systems programming	Unix Server	Embedded	Server (ML training)
Type	Library	Synthetic	Applications	Synthetic	Applications
Quality Reputation	Low	Low	**High**	Low	**High**
Free (no cost)	✓	✓	$1000 ($250 academia)	✓	✓
Easy to Port	✓	✓	✓	✓	
Revision Frequency	None since 1991	None since 1988	**3 years**	Never	**1 year (planned)**
Number of Programs	1	1	**10 SPEC89 - 23 SPEC2017**	1	**7**
Organization to Evolve Benchmark	✓		✓	✓	✓
Single Summary Score	FLOPS/second	Speed Ratio	**Geometric Mean Speed Ratios**	Speed Ratio	Weighted Mean Ratios **+ Std. Dev.**
Developer	Academia	Academia	**Academia & Industry**	Industry	**Academia & Industry**

Learning from Benchmark History (cont'd)

- Incorporate good ideas and avoid mistakes of past benchmarks
- Next learn from features of **not** widely quoted benchmarks that are candidates for embedded computing:

	EEMBC	*MiBench*	*BEEBS*	*TACLeBench*
Year	1997	2001	2013	2016
Initial Target	Embedded	Embedded	Compiler	Worst Case Execution
Type	Small Programs	Small Programs	Small Programs	Small Programs
Quality Reputation	?	?	?	?
Free	$20,000	✓	✓	✓
Easy to Port	No	✓	✓	✓
Organization to Evolve Benchmark	✓	No	No	No
Number of Programs	**41**	**36**	**80**	**52**
Iteration Rate	Rare	Never	**2 years**	Never
Single Summary Score	No	No	No	No
Developer	Industry	Academia	**Academia & Industry**	Academia

7 Lessons for Embench (Embedded Benchmark)

1. **Embench must be free**
   a. If behind paywall, not going to be as widely used
   b. Academics publish frequently and can promote the use of benchmarks, so free accelerates their adoption and hence publicizes benchmarks to the rest of the community
2. **Embench must be easy to port and run**
   a. If very difficult or expensive to port and run, then not as widely used
   b. Linpack, Dhystone, and CoreMark don't have good reputations yet widely reported presumably because they are free and easy to port and run
3. **Embench must be a suite of *real* programs**
   a. Real programs are much likely to be representative than synthetic programs
   b. A realistic workload is easier to represent as a suite of programs than as a single program
   c. Compilers and hardware change rapidly, so benchmarks need to evolve over time to deprecate pieces that are newly irrelevant and add missing features that speak to current challenge
   d. A single program is hard to evolve
   e. A suite of ≈20 *real* programs means a more realistic workload that is much easier to evolve

7 Lessons for Embench (cont'd)

4. **Embench must have a supporting organization that maintains its relevance over time**
    a. Need an organization to evolve suite over time
    b. Goal of refreshes every two years once we get to Embench 1.0
    c. Set up inside an existing organization, such as the CHIPS Alliance or the Free and Open Source Silicon (FOSSi) Foundation, rather than create a new org
5. **Embench must report a single summarizing performance score**
    a. If no official single performance score, then it won't be as widely reported
    b. Even worse, others may supply unofficial summaries that are either misleading or conflicting
    c. Might include orthogonal measures, like code size or power; each would have a summarizing score
6. **Embench should use geometric mean and standard deviation to summarize**
    a. Recommend first calculating the ratios of performance relative to a reference platform
    b. Report geometric mean (GM) of ratios + geometric standard deviation (GSD)
7. **Embench must involve both academia and industry**
    a. Academia helps maintain fair benchmarks for the whole field
    b. Industry helps ensure that programs genuinely important in the real world
    c. We need both communities participating to succeed

Follow the MLPerf Game Plan?

- ML had no widely adopted benchmarks in 2017
- Small group of ML academics and practitioners started meeting January 2018
    - Representing 3 universities and 2 companies
- Rapid iterations by small group meeting face-to-face could hash out foundation of benchmarks for Machine Learning
- May 2018 call for participation to fully define and evolve ML benchmarks* †
    - Grew to 7 universities and 39 companies
    - MLPerf Training 0.5 deadline November 2018; MLPerf Training 0.6 deadline May 2019
    - MLPerf Inference 0.5 deadline July 2019
- In less than 18 months, MLPerf became a well established benchmark

*Patterson, D., G. Diamos, C. Young, P. Mattson, P. Bailis, G.-Y. Wei., May 2, 2018, MLPerf: A benchmark suite for machine learning from an academic-industry cooperative. O'Reilly Artificial Intelligence Conference.
†Young, C., May 3, 2018. Why Machine Learning Needs Benchmarks. *Computer Architecture Today blog*.

First volunteer Opportunity: Pick Embench logo!

`tiny.cc/Embench`

Follow the Agile Benchmark Philosophy of MLPerf

- Make an initial version 0.5
- Then collect experience and feedback before making more ambitious version
- Like MLPerf, we expect to go through Embench 0.6, 0.7, …
- Until we have refined a version good enough to call Embench 1.0

Embedded Target for Embench 0.5

- Embench 0.5 targets small devices: Flash (ROM) ≤64 KiB, RAM ≤16 KiB
- May expand to larger devices in later iterations, but small devices for Embench 0.5

Candidate Programs for Embench 0.5

- Expected finding good 20 candidates would be difficult
- But past efforts at embedded benchmarks (MiBench, BEEBS, TACleBench, ...) have made many free and open programs that are easy to port and run
- Chose BEEBS (Bristol/Embecosm Embedded Benchmark Suite) because
  - It involved both academia and industry
  - It was a recent effort (2013)
  - It already borrowed good programs from MiBench and many others
- Reduce from 80 BEEBS programs to ≈20 Embench programs after looking at
  - Application categories covered
  - Relative emphasis on computation, memory, and branching
  - Size of the programs
  - Run times

Embench 0.5 Candidates

Name	*Comments*	*Original Source*	*C LOC*	*code size*	*data size*	*time (ms)*	*branch*	*memory*	*compute*
aha-mont64	Montgomery multiplication	AHA	151	1,052	24	4,000	low	low	high
crc32	CRC error checking 32b	MiBench	101	230	1,024	4,013	high	med	low
cubic	Cubic root solver	MiBench	121	2,466	192	4,140	low	med	med
edn	More general filter	WCET	285	1,452	1,600	3,984	low	high	med
huffbench	Compress/Decompress	Scott Ladd	309	1,628	1,004	4,109	med	med	med
matmult-int	Integer matrix multiply	WCET	175	420	1,600	4,020	med	med	med
minver	Matrix inversion	WCET	188	1,076	168	4,003	high	low	med
nbody	Satellite N body, large data	CLBG	167	708	656	3,774	med	low	high
nsichneu	Large - Petri net	WCET	2,676	15,036	8	4,001	med	high	low
picojpeg	JPEG	MiBench2	2,182	8,022	1,196	3,748	med	med	high
qrduino	QR codes	Github	936	6,056	1,540	4,210	low	med	med
sglib	Simple Generic Library for C	SGLIB	1,844	2,316	800	4,028	high	high	low
slre	Regex	SLRE	506	2,422	126	3,994	high	med	med
st	Statistics	WCET	116	880	64	4,151	med	low	high
statemate	State machine (car window)	C-LAB	1,301	3,686	64	4,000	high	high	low
ud	LUD composition Int	WCET	95	702	0	4,002	med	low	high
wikisort	Merge sort	Github	866	4,208	3268	4,226	med	med	med

Supplementing BEEBS

- We then discussed the important categories missing from BEEBS and added the best examples of free and open versions that we could find
  - Could use help with finding good code for elliptic curve DSA and bluetooth le decode

Name	*Comments*	*Original Source*	*C LOC*	*code size*	*data size*	*time*	*branch*	*memory*	*compute*
nettle-aes	AES encrypt/decrypt	Nettle	1,018	2,874	10,546	3,988	med	high	low
nettle-sha256	SHA256 digest	Nettle	349	5,558	520	4,000	low	med	med
elliptic curve DSA		*TBD*							
bluetooth le decode	Decode Bluetooth low energy encryption advertising packet	*TBD*							

Benchmark Score and Reference Platform

- Summary score is geometric mean (GM) and geometric standard deviation (GSD) of performance relative to reference platform for programs of benchmark suite
  - Ideal reference platform is widely available and well-known
  - GM and GSD guarantee consistent results even if reference platform changes, as happened with SPEC CPU benchmarks
- We tentatively chose the PULP RI5CY core as the initial reference platform
- Will collect and report results using `www.embench.org` web site

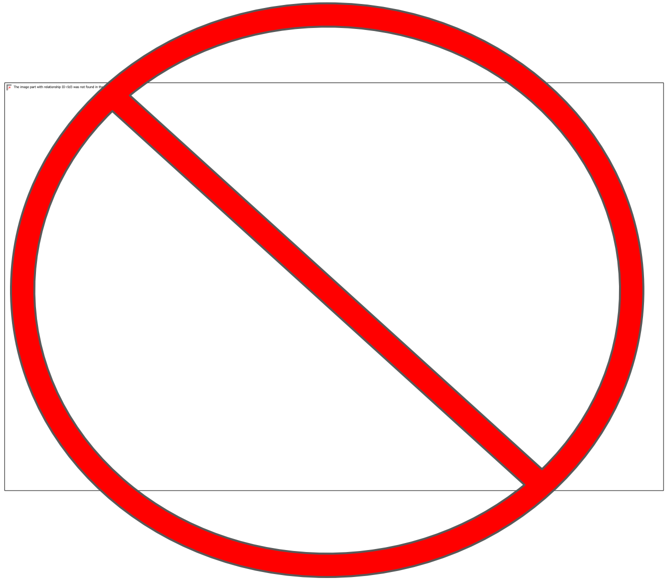
Report Code Size as well as Performance

- Code size is critical for embedded computing
  - Cost of memory for code can be a significant fraction of the cost of IoT device
- Will also report GM and GSD of code size relative to reference platform
- 0.9 score means the program is on average 10% smaller than the reference
  - (We think inverting the metric so that bigger score means smaller code would be confusing)
- We think code size is necessary for IoT yet novel to be part of formal benchmark

Report Interrupt Latency and Context Switching Time as well as Performance and Code Size

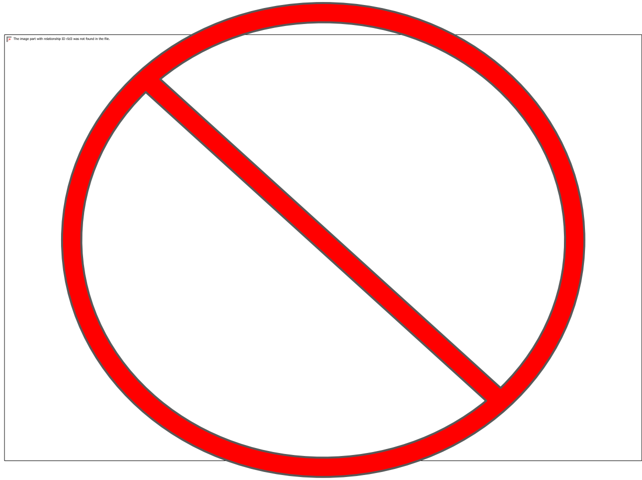
- Context switch time important for IoT devices
  - Active running low priority task but must switch quickly to higher priority task
- Interrupt latency important for IoT devices
  - Idle waiting for event to trigger into action
- For Embench 0.5, run a small assembly language program and measure its performance for switching contexts (rewrite for different ISAs)
- Interrupts will require a lab setup to measure latency
  - *Need help to design and document how to setup and run the interrupt measurements for interrupt latency to be included in Embench 0.5*
- We think context switch time and interrupt latency are necessary for IoT yet novel to be part of formal benchmark

Note: No Floating Point (Yet)



- No floating-point intensive programs in version 0.5 because:
    1) Floating point unit (FPU) significantly accelerates FP programs, but FPUs can be expensive for IoT chips
    2) Many embedded applications do little or no floating point
- We'll consider adding a set of floating-point intensive programs in a later iteration of Embench
    - Might have separate integer and floating ratings, like SPECint and SPECfp

Note: No Power (Yet)



- We rejected power for Embench 0.5 because:

1) SPEC Power has 33-page manual to fairly set up and measure power, including restrictions on altitude and room temperature. Which apply to IoT?
2) IoT devices are SoCs vs microprocessors. How compare performance per watt fairly as SoCs vary in complexity beyond their processors and memory?
3) Will want to test soft cores. How to run power experiment on soft cores?
4) Energy efficiency is often highly correlated with performance. Will results be interesting even if we did all the work to benchmark IoT power?

Example results: GCC RV32, LLVM RV32, GCC RV32E

	Reference: PULP RI5CY/GCC		*PULP RI5CY/LLVM*				*PULP RI5CY/GCC EABI*			
	Time (ms)	*Code size (Bytes)*	*Time*	*Ratio*	*Code size*	*Ratio*	*Time*	*Ratio*	*Code size*	*Ratio*
aha-mont64	4,000	1,052	5,271	0.70	1,112	1.02	4,799	0.83	1,194	1.13
crc32	4,013	230	4,159	0.96	242	1.06	4,162	0.96	226	0.98
…	…	…	…	…	…	…	…	…	…	…
Geometric Mean				0.99		1.11		0.92		1.07
Geometric St Dev				1.21		1.11		1.14		1.07
Lower bound of 1 St Dev				0.82		1.00		0.81		1.00
Upper bound of 1 St Dev				1.20		1.23		1.05		1.14
Context switch time		--			--	--			--	--
Interrupt latency		--			--	--			--	--

Call for Participation (info@embench.org)

- After good start, need and want help to complete version 0.5 (like MLPerf)
- We'll hold monthly meetings (including remote participants)
- Hope Embench 0.5 finalized in time to begin collecting and reporting results before the end of the year (like MLPerf did)
- If you have the time and interest in helping, please send email to info@embench.org
- 1st chance to help pick Embench logo: **tiny.cc/Embench**

Embench Conclusion

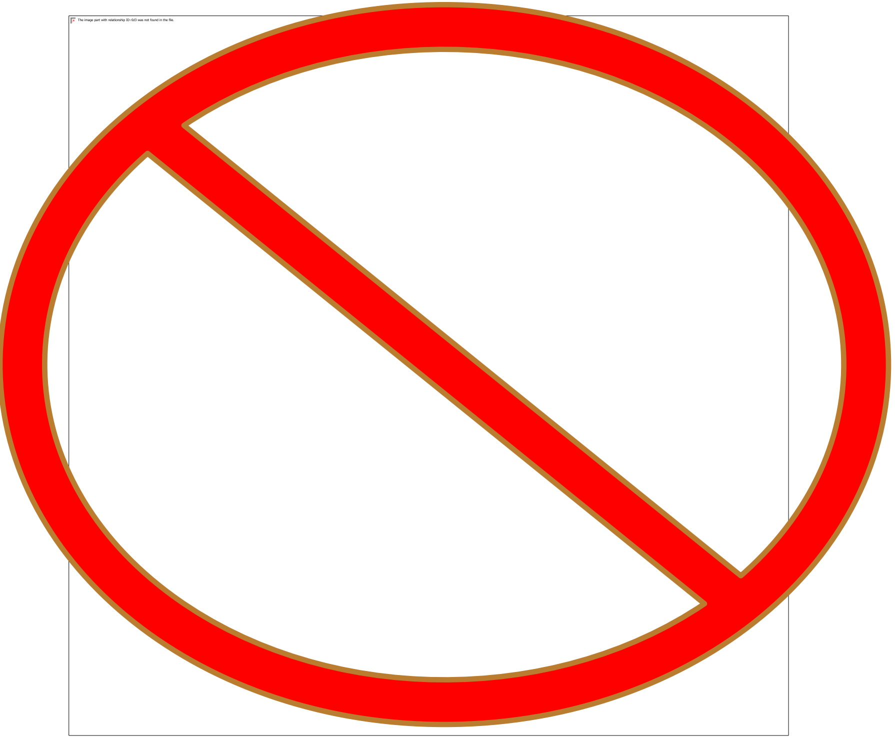
- Free
- Easy to Port
- Suite of 20 Real Programs (vs 1 Synthetic Program)
- Geometric Mean & Geometric Standard Deviation of Ratios to Reference Platform
- Also Report Code Size, Context Switch Time, and Interrupt Latency
  - Necessary for embedded IoT devices yet novel part of formal benchmark
- Sustaining Organization involving Academia and Industry to Evolve over Time
- Follows Agile Benchmark Development Philosophy: Versions 0.5, 0.6, …
- Given current state of widely reported benchmarks for embedded computing, we believe Embench—even the 0.5 version—will be a big help to the IoT field
- We want your help to finish and evolve Embench: `info@embench.org`

The image part with relationship ID rId3 was not found in the file.

RIOS (“Ree-Oss”) Laboratory at Tsinghua-Berkeley Shenzhen Institute (TBSI)

- RISC-V International Open Source Laboratory
  - 5-year mission to help raise the RISC-V ecosystem to the state-of-the-art (“uncore” IP)
  - A nonprofit organization that measures success by technology transfer
  - Produce industrial strength IP protected against patent lawsuits (e.g., follow expired patents or papers published 20 years ago): full time staff, access to Berkeley and Tsinghua legal scholars
  - Suggestions or interest: email `rios@sz.tsinghua.edu.cn`
- TBSI: Tsinghua-Berkeley Shenzhen Institute
  - Tsinghua University and UC Berkeley joint venture located in Shenzhen established 2014
  - Teach RISC-V, open source grad courses at TBSI to create future leaders of technology
- Director: UC Berkeley Professor David Patterson in Berkeley
- Co-Director: Dr. Zhangxi Tan in Shenzhen (Berkeley and Tsinghua alumnus)
- Co-Director: Tsinghua Professor Lin Zhang in Shenzhen
- Distributed lab with majority of ≈50 fulltime engineers at TBSI
- Looking for academic and industrial collaborators

Patent Troll Proof Intellectual Property



Rios is Spanish for “rivers”

Rivers name symbolizes collection of resources from many lands to create a strong force that changes the landscape, like RIOS lab at TBSI will do for RISC-V and information technology landscapes

First volunteer Opportunity: Pick Embench logo!

`tiny.cc/Embench`