MIT CSAIL Alliances | Arvind Export 3

Welcome to MIT'S Computer Science and Artificial Intelligence labs alliances podcast series. My name is Steve Lewis. I'm the assistant director of Global Strategic Alliances for CSAIL at MIT. In this podcast series, I will interview principal researchers at CSAIL to discover what they're working on and how it will impact society. Professor Arvind is the Johnson professor of Computer Science and Engineering at MIT.

In Arvind's research group, the Computational Structures Group, their mission is to enable the creation and development of high performance reliable and secure computing systems that are easy to interact with. The group is currently conducting research in the areas of computer architecture, hardware synthesis, computer security, and VLSI design.

In 2003, Arvind co-founded Bluespec Inc, a semiconductor design company which produces a set of tools and IP that remove barriers for developers. Arvind is a fellow of the IEEE and ACM, and a member of the National Academy of Engineering and the American Academy of Arts and Sciences. So Arvind, talk about your involvement in this century as you said with compilers or programming languages.

So I designed a programming language called-- always with others, a language called Bluespec. And Bluespec is really a language for designing hardware. So the compiler for Bluespec text is high level description and actually generates circuits from it. It generates actually Verilog for it, which can be further compiled using commercial tools.

And is it being used in industry today or is it open source?

Yeah, it's used. So I think Bluespec is also a company which has been going for almost 20 years, [LAUGHS] and it has had all kinds of very reputable TRA company customers. But it hasn't caught on in the sense people do specific projects using Bluespec, that's how it survives. But it's not broadly used.

Do folks like Intel and AMD, they use their own design tools?

That's true, but Intel certainly has used a lot of Bluespec for various projects. [LAUGHS]

What are some of the advantages of Bluespec?

I mean, one thing that you can see right away is most design today is done in Verilog. And we generate Verilog from Bluespec. So Bluespec is a much higher level language, it's much easier to write Bluespec, it's much better for verification than low level description languages like Verilog.

And I think those are the main advantages. And I continue to work on that. So I'm working on-- my dream is to-- that when you buy a microprocessor or other piece of hardware, it will come with a proof that it actually works, formal proof.

So are you working at all with Adam Chlipala in--

Yes--

--his--

--it's a joint project with him.

I see, interesting.

So Adam and I work together on the-- I've connections with programming languages folks.

Yes. And the whole idea is that you're basically verifying the security and integrity of the chip or the design.

Not-- I wouldn't use the word security integrity, though they are clearly affected by what I do. I'm verifying the functional correctness. So if you say that I have a RISK V processor, I would really like to give you a certificate with a formal proof indeed this is a RISC V processor, the design of it is--

I see.

So if you really are trying to build a highly trustworthy stack correctness thing, then these things will be of extraordinary use.

Let's shift gears a little bit and talk about flash storage. So in your mind, what are the biggest developments in flash in the last 5 or 10 years?

So flash has primarily been used for storage, like replacement for hard disks, rotating disks. So when it came out, it was significantly more expensive than rotating disks, but-- I mean, expensive in terms of cost per bit. But it was much faster, much more compact, much cooler, much more reliable. So it was offering all these advantages.

And Samsung did this very, very clever thing that even though underlying technology is completely different, they packaged it in such a way so that instead of using your hard disk, you could use SSD, solid state disk. And that marketing decision paid huge dividends. So for users, it was painless. They just had to decide were they're willing to spend more money in order to get better performance, et cetera?

And as the drive of price of flash dropped, it has just taken over. So now, almost all accessible storage is being done in terms of flash, and hard disks are used primarily for cold storage. So you still use it, but data that is rarely accessed, you'll put it there. Any data that you really want to access is all being stored in flash now.

And what are some of the advantages of flash over DRAM?

So they're orthogonal. So first, let me give you the advantages then I will tell you the limitations or disadvantages. So the biggest advantage is flash density is orders of magnitude higher. I mean, you talk of packing 2.5 terabytes in less than 2 inches. So this bigger disk and 2 terabytes, I mean, that's humongous amounts of data.

It's extraordinarily reliable, and it consumes very little power. And it's much, much faster than rotating disks, than hard disks. OK, so these are some of the advantages. The disadvantages, it's access time is much slower than DRAM, not by 10%, like three or four orders of magnitude slower. So in other words, different places in the system. So your main CPU is always accessing DRAM while it's being backed up by flash storage because it takes time to go there and come back.

So really, it has been a replacement for storage but not a replacement for DRAM. Now, there are other people, other companies and newer-- as they are known as, NPR'S nonvolatile memories like ReRAM and Dram, and there are many, many technologies that are on the market which want to be very close to DRAM, they want to compete with DRAM.

But commercially, I don't think there is a success yet in that. Because as you increase the speed of it, it's density decreases a lot. So unlike flash which had a clear marketing point, these other non-volatile memories-- a lot of work is going on in many companies. In my opinion, none of them have seen enough of a commercial success because they haven't found the right place to put it in the system. Because if you created yet another hierarchy, it complicates the system.

How has your research contributed to flash or these technologies around flash?

OK, that's a very good question. So before I go there, let me just say one or two more things about flash. So the progress in flash technology itself is dramatic. It's the first semiconductor technology where three dimensional chips have been extraordinarily successful.

So when you're buying a flash today, it has 32 to 48 layers in the chip itself. And I think in the very near future, it will probably go to over 100, maybe 100 to 228. And on the research side, people are talking about technologies which will be even more than 128, so you may go to hundreds and hundreds of layers. So that's just-- that tells you that it's possible to build extremely dense storage. And that's why everybody else is having a hard time to catch up.

And what has led to that innovation? What has led to being able to put more layers on a chip?

I think that is the real circuit size, circuit innovations and I'm not an expert in that. So lots of things are known about that, but just as we use some technology for processors and slightly different technology for DRAM. Even though fundamentally, they are the same because DRAMs, you optimize for something. So similarly, this is also semiconductors, but you're optimizing it differently because that market is so important.

And the moment you have the market, you have money to do the research, continuous research. And any time when you get ahead in this game, others are left gasping. So I think that's one reason. The second thing is what people do not know is even the tiny pen drive you buy which has flash memory inside it, it's really a full blown system. So it has a four processor arm core inside it. [LAUGHS]

And it has several gigabytes of DRAM inside it. This is needed just to manage the flash storage. So when we talk of flash storage, we talk of storage, but it's really a system. There is a whole load of stuff going on there and that's just innovation over on top of innovation. So what made me interested in that was I wanted to see if I can do more computing in flash.

Because when I saw this thing, I mean, I remember somebody came from Samsung, some vise president, and he's showing me this box, this is like six years ago, say, cigar size, cigar box size thing. And I can't remember now how much storage it has, but probably it had 32 gigabytes of storage, I can't remember. Some humongous amounts of storage in that.

But what was shocking to me is it had 14 multi-core arm chips on it. And it had a lot of DRAMs, I'm saying. I should be able to do more computing on this and can I do something to the organization of flash so that they need less resources? So both those problems have been of great interest to me.

So you're saying like the thumb drives that you buy at best Buy for $9.99 for 64 gigabytes are a computing platform?

That's right. They're not sold like that. I mean, you think of it as storage, but if you open it up, you say, there is so much stuff here. Why can't I compute on that? So that was the hardware motivation. Now, what problems? So for example, I can talk of several different problems. But has the easiest one to relate to is genomics. So if we ever get into personalized medicine, then genome-- your genome is going to play a very important role.

And you may want to know, for example, mutations in your genome, cancerous, tumors, and so on, that's enormous amounts of computation. I look at your genome and I look at the reference genome or your cancerous genome versus your normal strings and I want to compute very fast and I want to compute very cheaply in doctor's office or maybe at home.

So you can imagine that I can program all that on the flash drive itself. So when you go and buy this stuff, even though you're buying an SSD, I don't sell it to you as an SSD. I'm saying, oh, this is your genome assistant or your tumor detector or something like that. So I've been looking at many problems like that-- not many, I mean, several problems of that sort.

Where you can bring down the cost of computing, these problems normally will be done on supercomputers, big machines. And I'm saying we can do them on very small machines provided they have enough flash storage. So just imagine your normal computer connected to the flash, except that this flash, I'm doing a lot of computing near flash.

So it's a coprocessor?

It's a coprocessor. And near flash computing is lots and lots of problems are amenable to that. So how is the genome problem done today? Of course, they have to also store the genomics data on flash because it's too much. And then when they want to compute, they suck it in to your DRAM.

You never have enough DRAM, so you start putting multiple machines so that collectively they have enough DRAM. And then you operate it in that fashion. And that's the part I don't like. I want to compute externally, to keep it cheap, even if it's slightly slower because I can make it much, much cheaper and simpler doing it like that.

And how do you go about that? Is it the design of the circuit?

It's a combination of hardware and software. So there is hardware assist needed, some special operations, plus software is needed because often, you have to change the algorithms, you have to adjust the algorithms. Actually, I will use the word external algorithm, which actually goes back to 50s.

So the first problem people faced where there was a lot of data and you wanted to compute with it but it won't fit in the memory was sorting. Because IBM was selling all these machines to insurance companies and so on, banks, and you want to sort humongous amounts of data on customers.

So they develop the techniques which are called external sorting. So instead of bringing all the data into DRAM or core and sorting it, you say, let me just keep writing it back on the tape drive or disk and I can sort that way also. So these are different class of algorithms which are known as external algorithms. And once you understand it, there is nothing magical about it. I mean, you can develop it. Not every problem is amenable to external algorithms but many are.

Is it problematic for flash in the number of read, write cycles or is innovation being made to overcome that limitation?

I think that, of course, is a limitation. Practically, turns out to be not a limitation in the way flash is used. On the other hand, if flash-- people start using flash the way I'm using it, for computing, then that will become more serious issue because you will have a machine whose capacity decreases as you use it because parts of it get worn out and you can't use it. So yes, there is a lot of research going on. I hear almost magical results sometimes, but nothing has come to fruition.

Let's change gears a little bit and talk about graph AI systems.

So graph AI is simply a problem where some constraints are represented in terms of a graph. So for example, I may have a map. So the roads by which it's connected, it forms an automatic graph. But cities are much more than that. I mean, there are huge feature vectors associated with the city-- what is the population, there is that-- depending on what you are interested in.

And different things may require different kinds of graphs. So graphs just represent some kind of constraint on data which is associated with an entity. So this data that is associated with an entity is often called a feature vector. And some aspects of the feature vector can be captured by a graph. And then you may want to decide more things about those features. You may want to extract, you may study something.

So it has so many applications. I mean, any problem can be formulated, actually, in terms of graph AI problems. So for example, suppose I'm into fraud detection. So I may build a graph of your social network. I go there and see who you are friends with, et cetera, and I may have a graph of financial transactions.

And I will see there are some transactions that look funny because I see a loop thing which is an indication that it's probably fraud, somebody is trying to cheat here. So that's what people do. I mean, they have existing data, and they study problems-- the study features of that graph and the features sometimes tell them the exact thing, it is indeed a fraud, or it narrows it down, it's probably fraud, something like that.

People do this with protein, protein interaction. So suppose I'm trying out new drugs. The very first thing I have to do, even before I can work on efficacy of it, does it do any harm? So you like to see similar drugs, what kind of effect do they have? So now you look at your thing and you are trying to see if that graph-- that older drug graph was like that, what is likely to be the places where you might be having a negative impact or stuff like that?

And similarly, you can study the efficacy of it too, what you are likely to-- So I think any problem can be formulated this way. And how you compute it, a lot of smarts are involved. From computation point of view, it all looks very similar. But algorithmically, what optimizations you may use here and what optimization you may use for some other problem are very, very different. So I think this is one area where verticals will make a lot of sense.

Let's talk about-- let's go back to computer architecture, let's talk about computer architecture. And so what major problems exist in computer architecture and what research is being done to address those problems in your opinion?

So for a long time, computer architecture was really about the design of processors, design of memory systems, et cetera, and we still do some of that, but that's not where the real action is. So the action moved to accelerators because they are much more power efficient. Because power is the biggest thing. And if I can compute the same thing using 1/10 of the power, then more applications become feasible.

So what people discovered immediately was that if you make the thing more special purpose, so if you take an algorithm which is doing fewer operations and you directly try to implement it in hardware, then sometimes you can get super efficient hardware. So that has become very, very important. And the biggest example of that right now is you see neural networks, so TensorFlow and this and that.

People are designing special purpose hardware, and those machines are only good for doing that kind of stuff. So a lot of action in that. And how to connect these things together, how should I think about it? How should I manage resources in these things? So these problems remain. Long term, there are problems at two ends in this.

So if you come to edge computing from your Fitbit to you name it, I mean, the amount of computation that goes on at the endpoints is so much-- it's so fascinating. I mean, whenever I go to a doctor I'm saying, this is crazy. I mean, I come here and he measures my blood pressure and sends me home, why is my blood pressure not being measured all the time and transmitted when something interesting worthwhile is happening in that?

So I think you will see lots and lots of changes like that. So this is all edge computing. No end to it, just limited by our imagination and markets. The other extreme is data centers. These are the size of computers we have never seen before. So in those places, efficiency is super important. I mean, dollar cost is very important.

And there is a lot of innovation going on there also. How to do it so that I can do the same computation, same speed, with half the resources. Because there was too much waste of resources. So people are analyzing everything, better resource management, and that often requires changes in the hardware itself.

Is there any other research from the Computational Structures Group that you think is particularly interesting or exciting?

Well, the other research, as I told you, is about this verification business. It's my dream to sell chips with proofs. [LAUGHS] I think that I'll make a big difference. I mean, that will be the ultimate trustworthy computing.

Do you think the general public is aware of the importance of this?

No. I think security questions are always much tougher for people to understand. And also generally, security is breached by such stupid things. You don't have to be so sophisticated about breaking it because people use stupid passwords or leave it lying around or they just write all the passwords in one sheet of paper and keep it somewhere. [LAUGHS]

So I think therefore, very, very secure computing is important, but you see, we have to assume that banks, et cetera, are safe, I mean, your financial transaction. I mean, that's why when you're browsing the web, there are clearly two phases. You can book whatever seat you want, but the moment money transaction is involved, it starts getting all encrypted.

This is just a recognition of the fact that encryption costs money so you don't do everything that way. And I think it'll remain like that. That very secure computation will remain expensive, and when we design new systems, we will pay attention to that, we will refactor it. We'll say, OK, this part has to be done very securely.

Other parts, we don't care, you can just use ordinary computing. It's really no different than the way we use water. You use bottled water for drinking, but you can use tap water or even worse, you can use water for watering the lawn, which is different.

Fascinating. Well, where can people go to find out more about your research?

They should still go to my web page. And after talking to you, I'll go and update them. [LAUGHS]

Excellent. Arvind, I thank you for your time, it's been a fascinating conversation.

Thank you very much.

[MUSIC PLAYING]

If you're interested in learning more about the CSAIL Alliance program and the latest research at CSAIL, please visit our website at cap.csail.mit.edu. And listen to our podcast series on Spotify, Apple Music, or wherever you listen to your podcasts. Tune in next month for a brand new edition of the CSAIL Alliances podcast and stay ahead of the curve.

[MUSIC PLAYING]