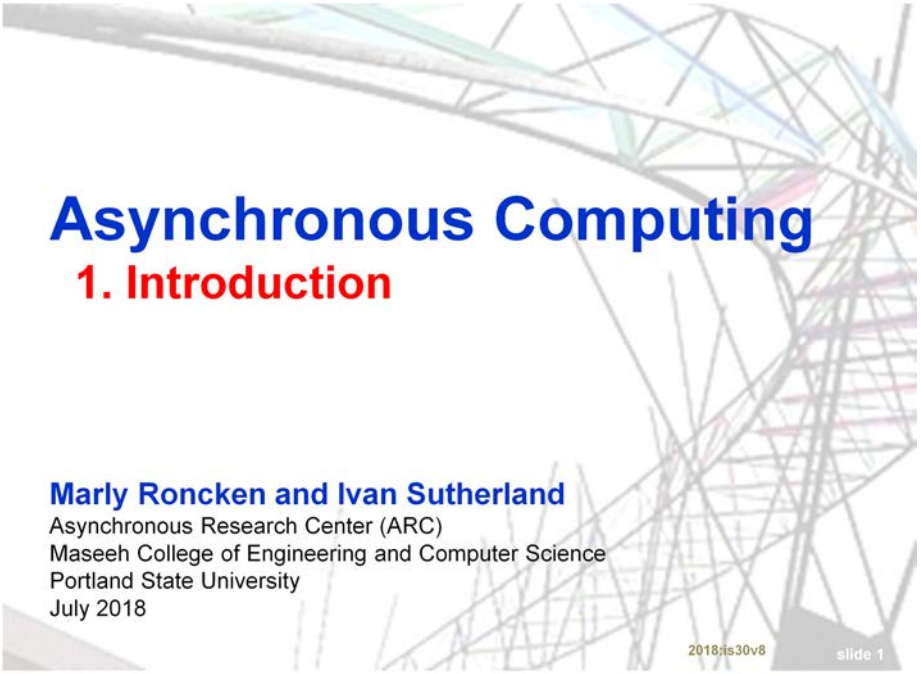


Good afternoon.
My introduction will set the stage for the rest of our talks.

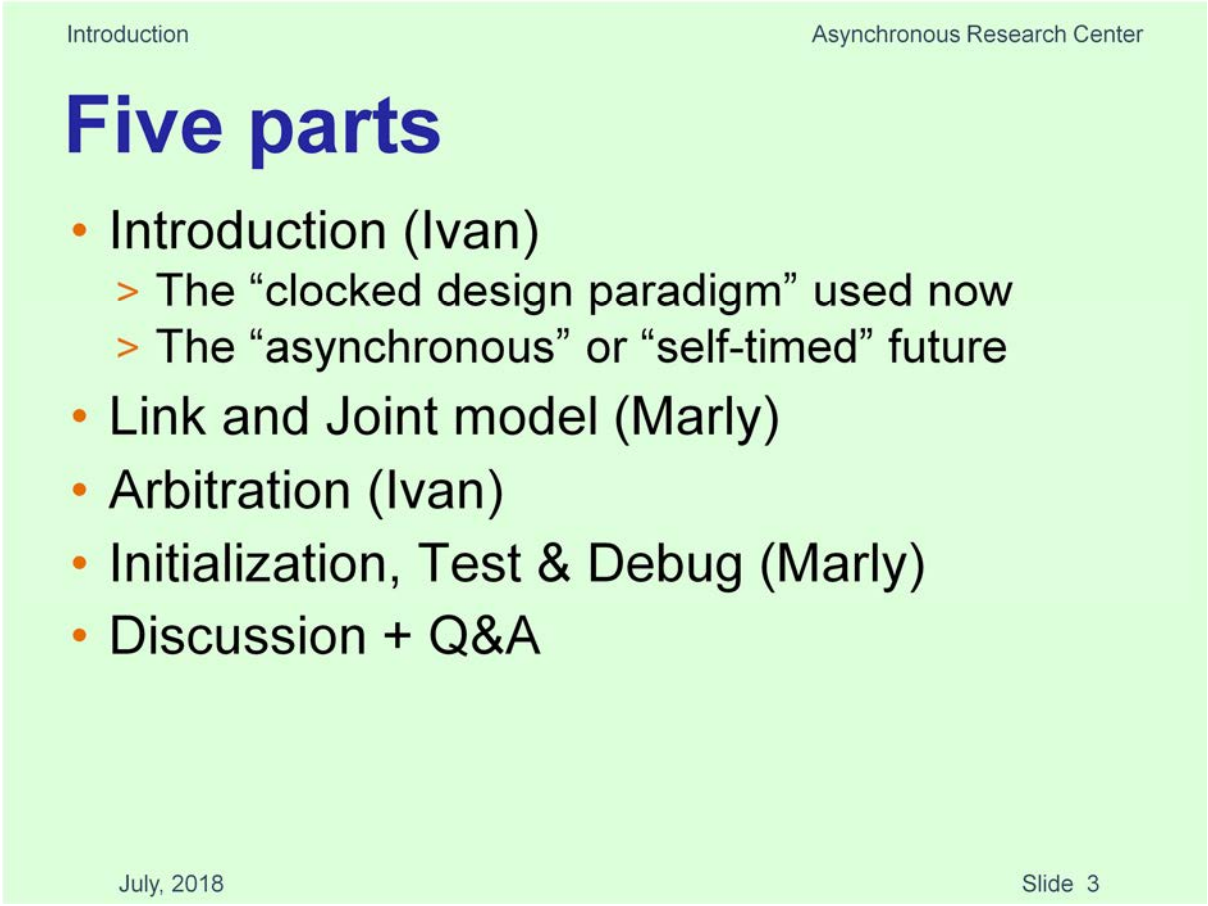
Chronos = time. Syn = same. A = not.
Asynchronous means “not at the same time.”

Marly and I started the Asynchronous Research Center (ARC) in 2009.

Marly Roncken and I married in 2006
and we do research together.

We worked separately on asynchronous systems,
Marly in The Netherlands, I in the USA.
We met at the asynchronous conference, ASYNC, in 1994.
She won the best paper award, I had given a keynote speech.

This slide shows some of the places we have worked.
I'm now 80 years old – it's impolite to ask a woman's age.

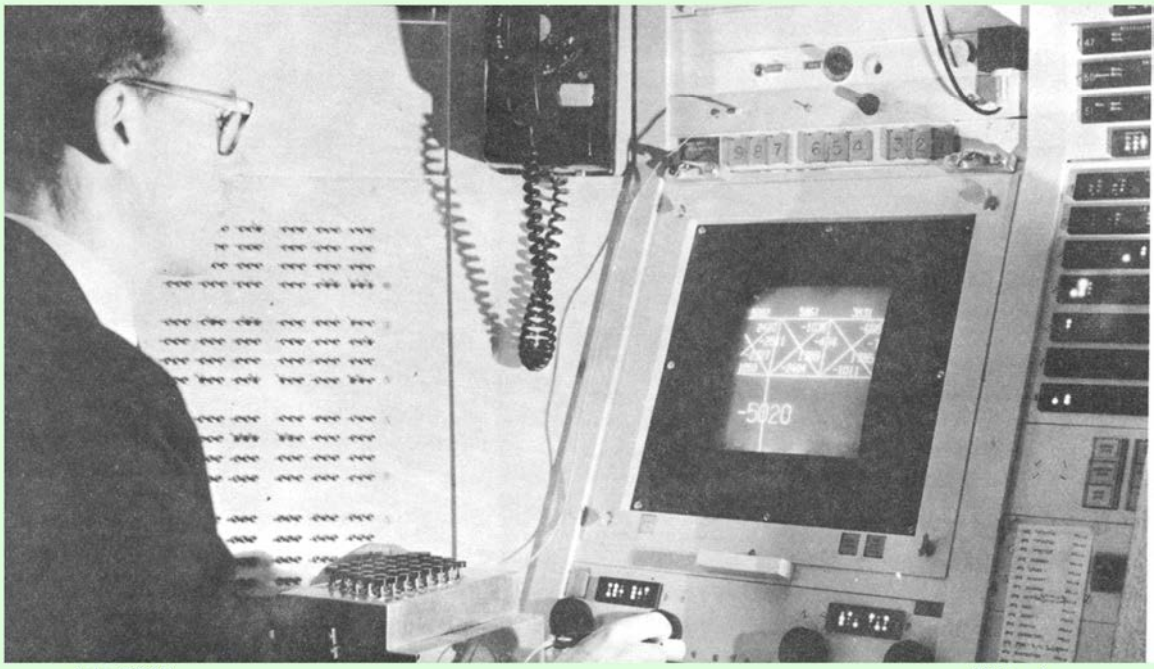


This ShanghaiTech Lecture has five parts.

My introduction describes today’s “clocked” design paradigm and the forthcoming asynchronous paradigm.

Arbitration is the only logic unique to self-timed systems.

Sketchpad (1963)



Fifty years ago as a young man I wrote Sketchpad.
(That may be why I was invited to come here today)
But our talks today are about another subject.

Marly and I will talk about a long overdue paradigm shift.
A Paradigm is how we think about and do things.
In USA we eat with a knife and fork.
In China you use chopsticks.

TX-2 computer (1958 – 1978)

100 KHz clock = every ten microseconds

Light goes 3 km in ten microseconds

Wires very much faster than logic

Can ignore Wire delay

I wrote sketchpad for the largest computer of its day, TX-2.

TX-2 filled a large laboratory.

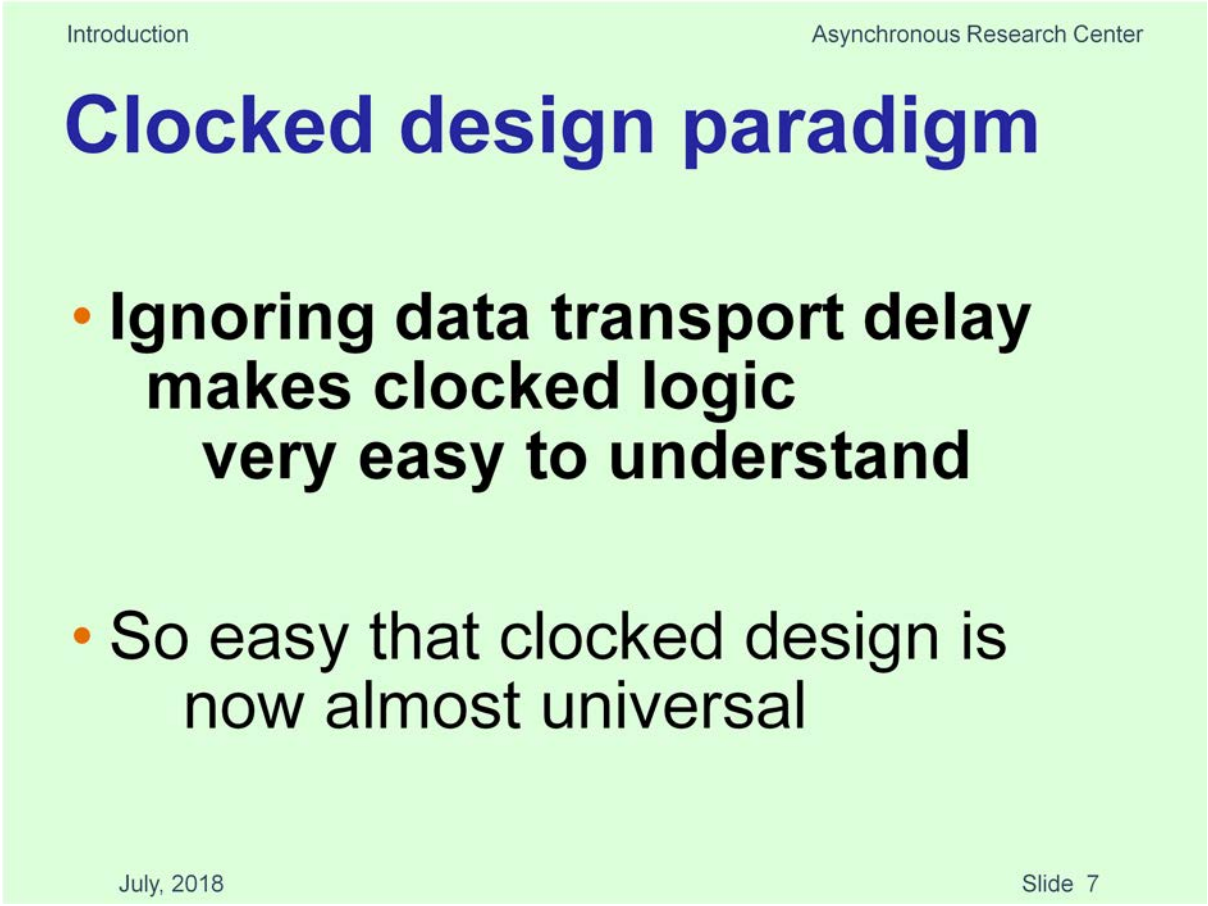
TX-2 ran at 100K operations/second,
one every ten microseconds.

In that much time light goes about 3 km.

Clocked design paradigm

- Logic acts on each clock tick
- Assume instant data transport to other logic
- Do next step on next clock tick
- All logic marches in step to the clock beat
- Step by step progress
- Very easy to understand

The clocked paradigm for logic design is almost universal. It is easy to understand because it ignores transport delay.



Clocked design is easy to understand

because it ignores data transport delay.

All Universities teach it.

All engineers know it.

Nearly all equipment uses a clock to pace its logic.

Chess: a clocked metaphor

Chess moves are like the steps of clocked logic.
Between moves all pieces are stable.
Each move is instantaneous.

Chess is an apt metaphor for a clocked system.

because pieces move instantly when they move.

Each move takes zero time.

Between moves the state of the board is stable.

Progress goes forward step by step.

What could be simpler?

Chess: a clocked metaphor

Question: who runs Faster?
A Horse (knight) or Chariot (rook)?

Let me ask you a question:

Which runs faster -

a Horse (that I would call a Knight)

or a Chariot (that I would call a Rook or a Castle)?

Chess: a clocked metaphor

Question: who runs Faster?
A Horse (knight) or Chariot (rook)?
The question is meaningless.

Which runs faster is meaningless
because they both move instantly to their next position.

Chess players

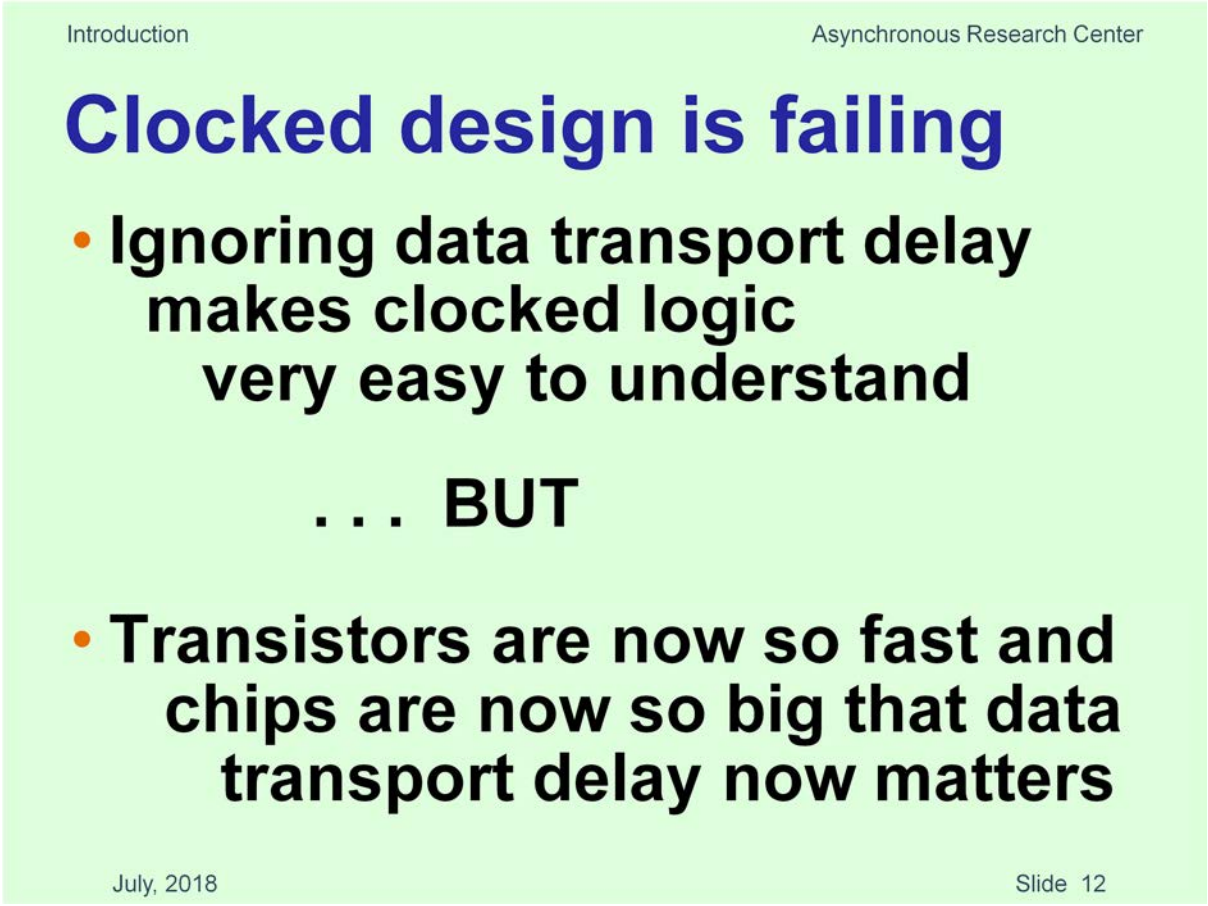
- Lack any notion of how fast pieces run because all pieces move instantly
- Lack a vocabulary of running speed
- Lack a way to reason about arrival time
- Strategy needs only **where** and not **when**

- Asking a chess player which piece is faster is like asking which digit is faster the digit “4” or the digit “7”

The result is that Chess players think only about position.

They think about WHERE but never about WHEN.

Strategy ignores WHEN.



Advances make transistors faster and chips bigger.

Now transport delay (that we used to ignore) matters.

That's good for electronics but hard on designers.

Their simple way of thinking must be replaced.

A new paradigm is coming

- Designers will no longer ignore delay
- Asynchrony (no clock) is inevitable
- Asynchronous = self-timed
- Paradigm shift is coming
- This session offers an early look
  - > For Computer Science and Circuit people
- We want ShanghaiTech to participate

Asynchrony is inevitable

because it conforms to Einstein's view of Physics.

Space and Time are intimately related.

We got away with ignoring transport delay.

But now we no longer can.

Football: a self-timed metaphor

Football is continuous – no marching in step
Spread out over area and time
Who arrives first matters a lot

I used Chess as a metaphor for clocked design.

Football is an apt metaphor for self-timed design.

It happens over area.

Each player is independent.

Each thinks and runs at his own best pace.

Delay matters: strategy must involve both where and when.

Football

- Football flows
- Split second decisions
- When and where matter
- Question: which is Faster? My team or your team?
  - > Great question – faster may win the game
- Who arrives first matters a lot
- Strategy must reason about when and where

Football players run as fast as they can.

How long it takes to get in position matters.

Transport delay matters.

Including transport delay makes strategy harder.

The title of my SSIST 2018 talk on Monday was
“STOP THE CLOCK.”

We must stop playing football with chess thinking..

Continuing to think chess when playing football is a bad plan. We can do better, and that's what we're here to talk about.

The self-timed paradigm

- Asynchronous = self-timed data transport
  - > Nearby arrives soon; further takes longer
  - > Say when data arrive
- Asynchronous = self-timed operations
  - > Easy is fast; hard takes longer
  - > Say when each action is done
- Use logic gates to schedule
  - > When to act
  - > When to transport data or fetch new data

Self-timed data transport announces when data arrive.

Self-timed operations say when they are done.

The self-timed paradigm does logic on these “done” signals to keep actions in sequence.

Need concurrent thinking

- Hard to think concurrently
- Takes a new point of view
- To understand a beehive
  - > It's not a chessboard
  - > Follow one bee at a time
  - > Think like a bee
- What does each bee do?

The clocked paradigm encourages us
to look at an entire system
and understand the steps it does.

Each step happens all at once everywhere.

But the world isn't like that.
Different things happen at different places and different times.

Looked at from the outside, a bee hive is hard to understand.
But the bees don't think so – each knows what to do.

Point of view: self-timed

- Every action reports when it's done
- Like a software subroutine return
  - > Carries an answer AND
  - > Allows next code to proceed
- Every data transport reports arrival

- A vocabulary for talking about WHEN lets us apply logic to sequencing actions rather than marching with everyone in step

A software subroutine returns more than an answer.

The "return" from a subroutine says that it's done.

That sort of reporting lets us use logic gates
to keep actions in their proper sequence.

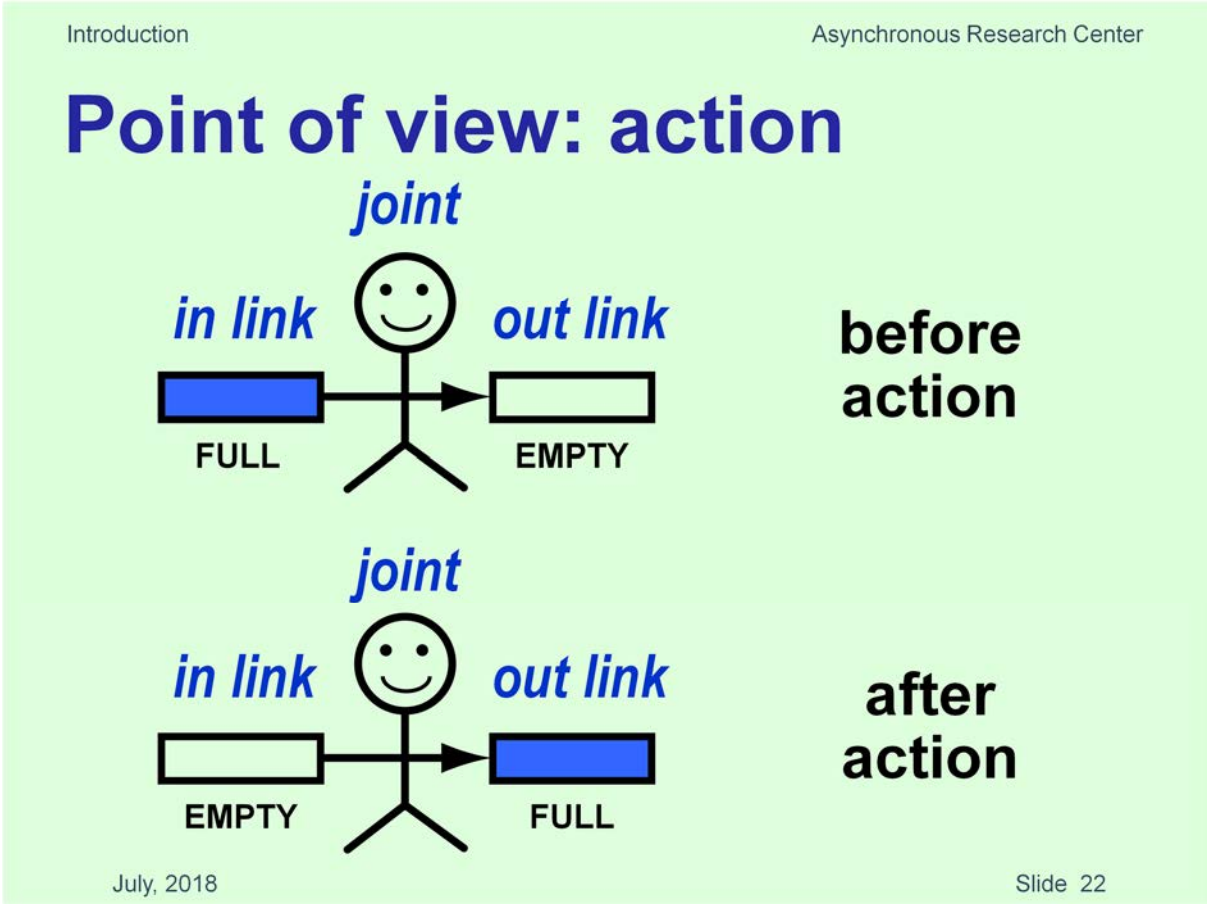
Point of view: data

FULL EMPTY FULL

THE MOST IMPORTANT THING ABOUT A PARKING SPACE
is whether or not you can park there: Is it FULL or EMPTY?

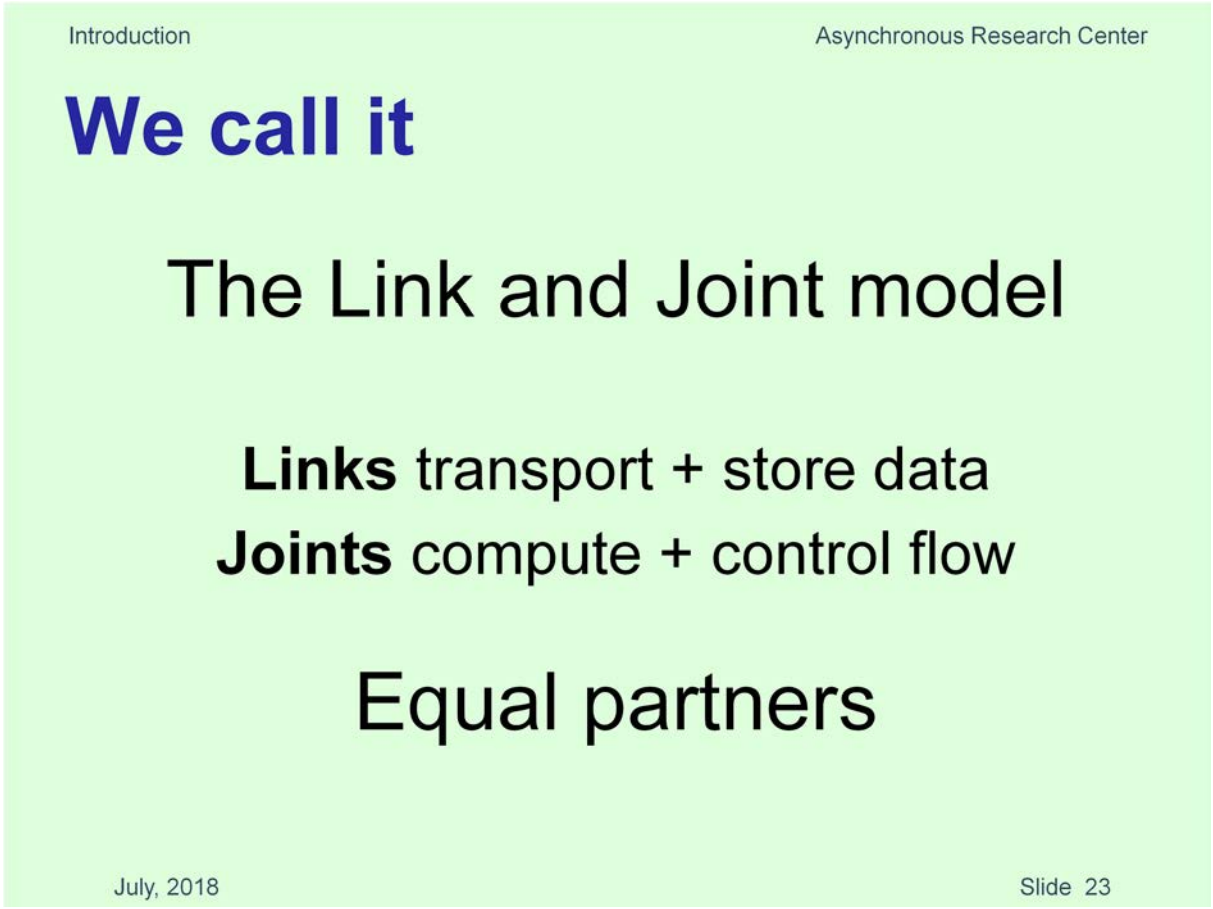
Ordinary computers don't know
whether their registers are FULL.
A 32-bit clocked register represents exactly 2^{32} states.
Only a programmer knows if a register holds meaning NOW.

When you start a program
do the values in main memory have meaning?
Or are they left over from an earlier user?



The full input Link is colored blue.
The empty output Link is blank, so the Joint can ACT
Doing so fills the output Link and drains the input Link.

Although only one input and one output Link appear,
Joints can serve many Links.



Links are mostly wires

with just enough logic
to tell whether a Link is EMPTY or FULL of data.

Joints are mostly logic

with local wires to connect their logic gates.

If you take only one thing from today's talks
remember that logic and communication
must be equal partners.

The Link and Joint model

- Roncken et al., *"Naturalized Communication and Testing,"* ASYNC-2015
- Roncken et al., *"How to Think about Self-Timed Systems,"* Asilomar Conference on Signals, Systems, and Computers, 2017

Marly invented the Link and Joint model.

She published it in ASYNC 2015.

She is going to say a lot more about it.