



ORACLE[®]

Verification Bug Metrics A Different Approach

Greg Smith

Greg.A.Smith@oracle.com

Talk Outline

- The hardest question for a DV manager to answer
- How does chip design use metrics today?
- How does Software do it?
- Adapting SW metrics principles to HW design
- Actual project data

Some Background

- Became a design verification manager at HP
- With technique I am presenting today, successfully taped out 10 ASICs with first pass success
 - Some data from those projects is presented here
- The hardest question for a DV manager to answer:

What do you think it is?

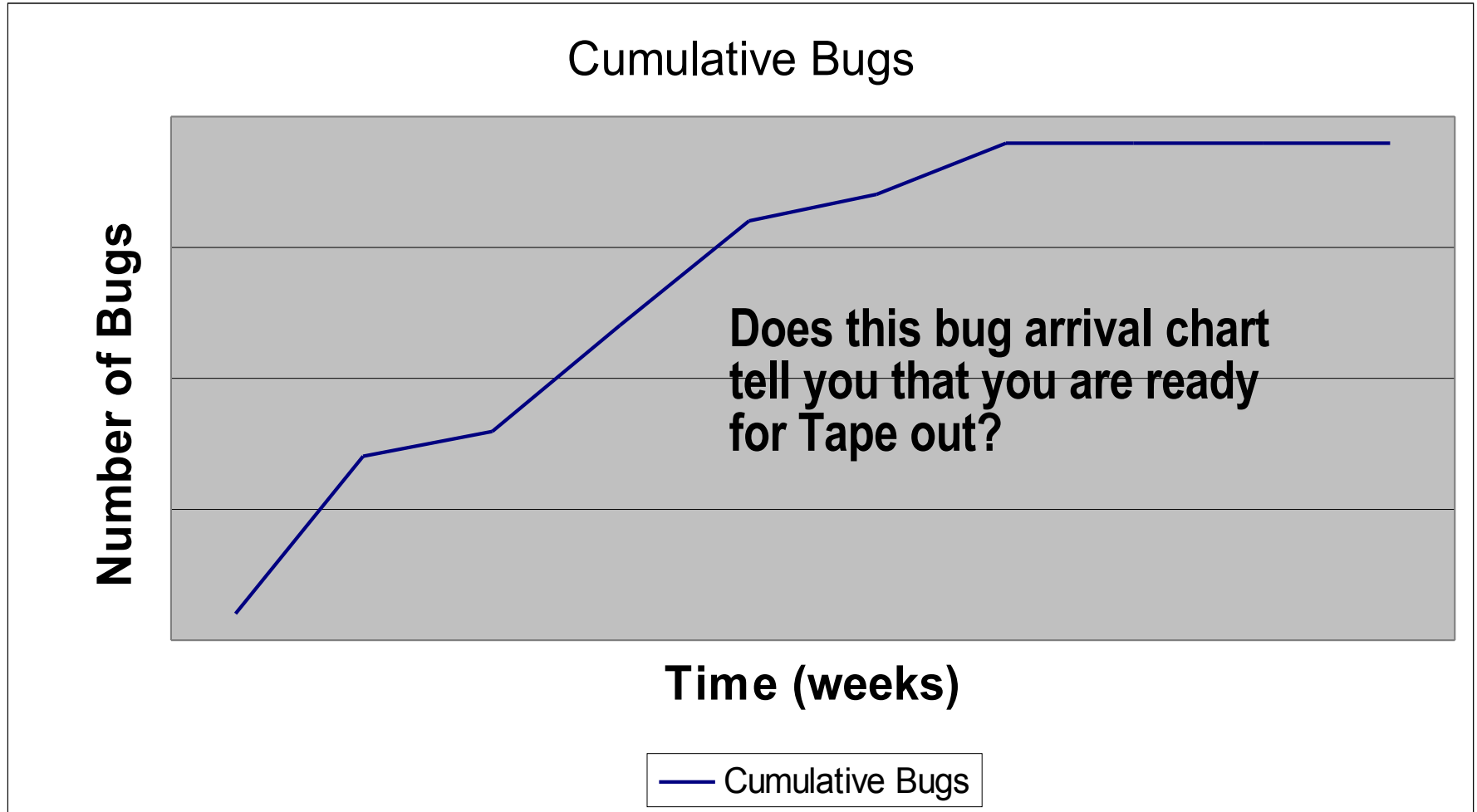
- The hardest question for a DV manager to answer:

When will you be done?

Metrics Used Today

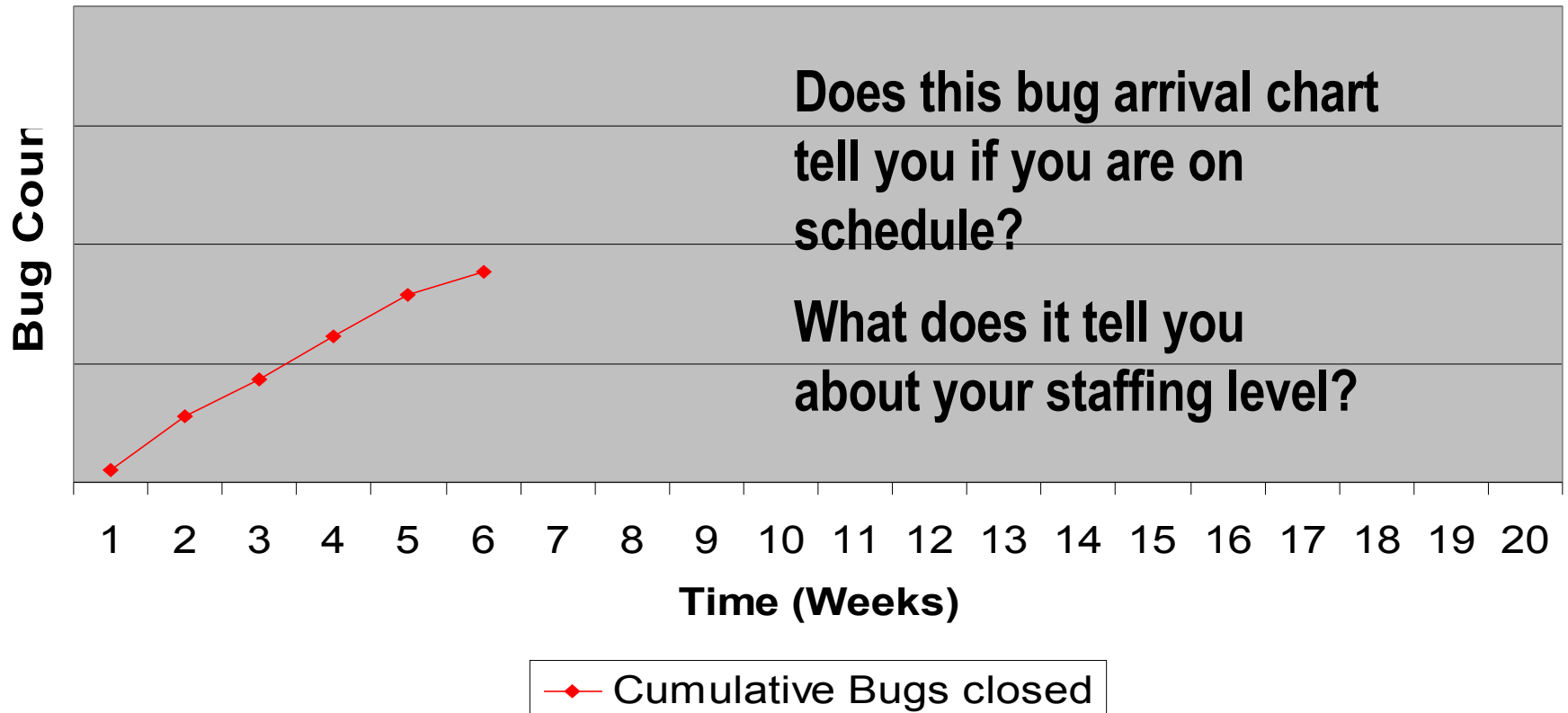
- What are the most common metrics in use today to track execution and measure progress:
 - > Completion of Test Plan
 - > Functional & Code Coverage percentages
 - > Bug Arrival Rates
 - > RTL rate of change
 - > Schedule milestones
- The above are all backwards looking and subject to human error of omission!
- I was searching for a quantitative, **calculated** way to predict and track schedule.

HP Project E



HP Project M

Cumulative Bugs closed



How Does Software Do It?

- Researched web for Software Development Metrics
 - Software +defect +reliability +metrics
- Amazing amount of material: PMI, SEI, University and government research
 - Reference Holly Richardson :“Rayleigh Curve Based Estimation”
- Discovered many corroborating papers and studies on related topics
 - How many resources are needed, when, and for how long
 - **Bug arrival curve predictions**
- Technique I am presenting today was used on the software project for the NASA space shuttle

The Rayleigh Distribution Model

$$E_m = \frac{6E_r}{t_d^2} \times t e^{-\frac{3t^2}{t_d^2}}$$

So what is all this?

E_m = errors expected in this period

t_d = Total # of measurement periods

t = elapsed time (This measurement period).

E_r = total # of errors expected

- Therefore you need 2 pieces of information:

Project Duration

of bugs expected

Applying the Principle to HW Design Project Duration (t_d)

$$E_m = \frac{6E_r}{t_d^2} \times t e^{\frac{-3t^2}{t_d^2}}$$

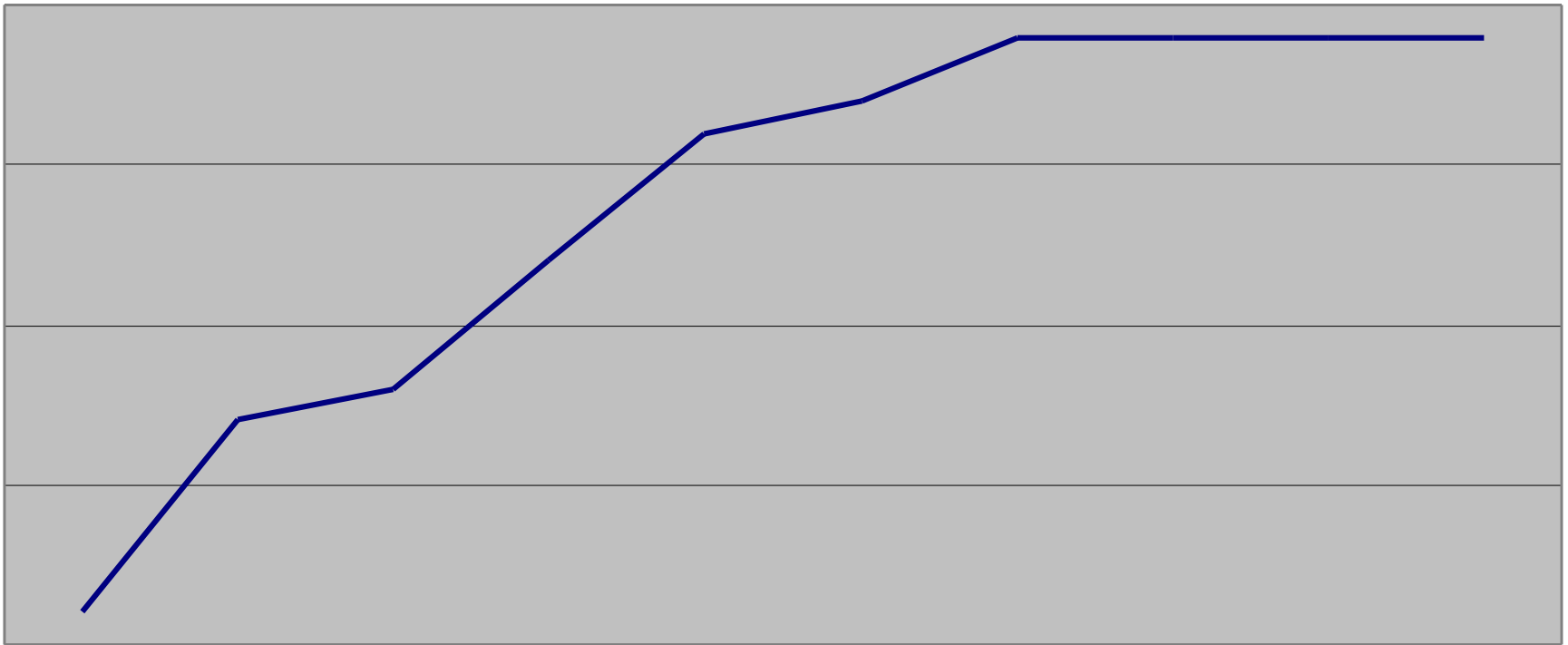
- # of measurement periods between when you start counting bugs and when you freeze or tape out or equivalent milestone
 - Typical would be weeks
- t term is the given week

Predicting the # of Bugs to be Found: What is your bug density? (E_r)

$$E_m = \frac{6E_r}{t_d^2} \times t e^{-\frac{3t^2}{t_d^2}}$$

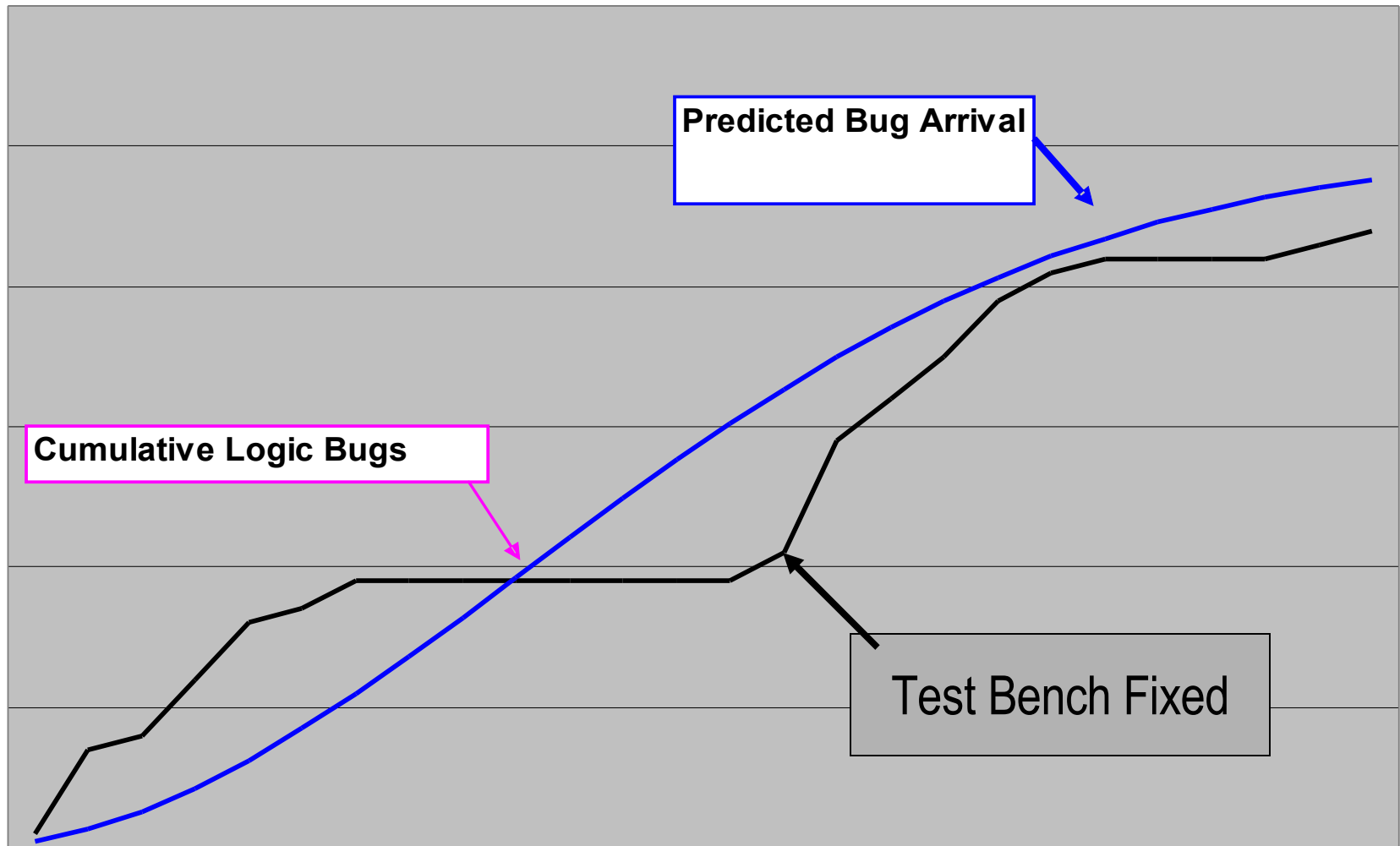
- How I went about figuring this out:
 - SW Uses bugs per LOC or KLOC
 - Went through bug data base for several past projects
 - Total up all RTL bugs
 - Sized the corresponding design (lines of Verilog code)
 - Observed a trend which established my initial “guess”
 - Our density was 1/150 LOC for all projects
 - SW norms are 1/50 – 1/100

Cumulative Bugs



— Cumulative Bugs

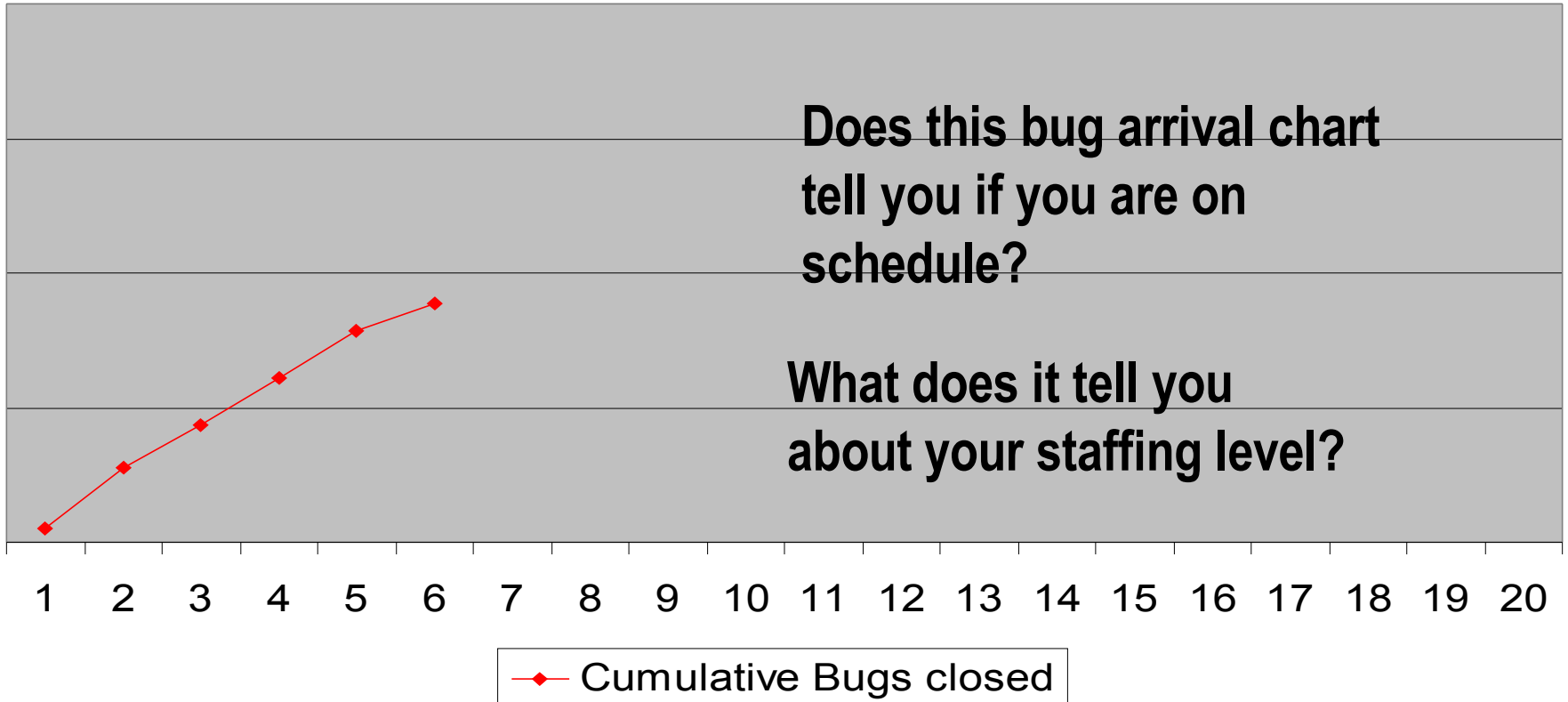
Something WAS Wrong



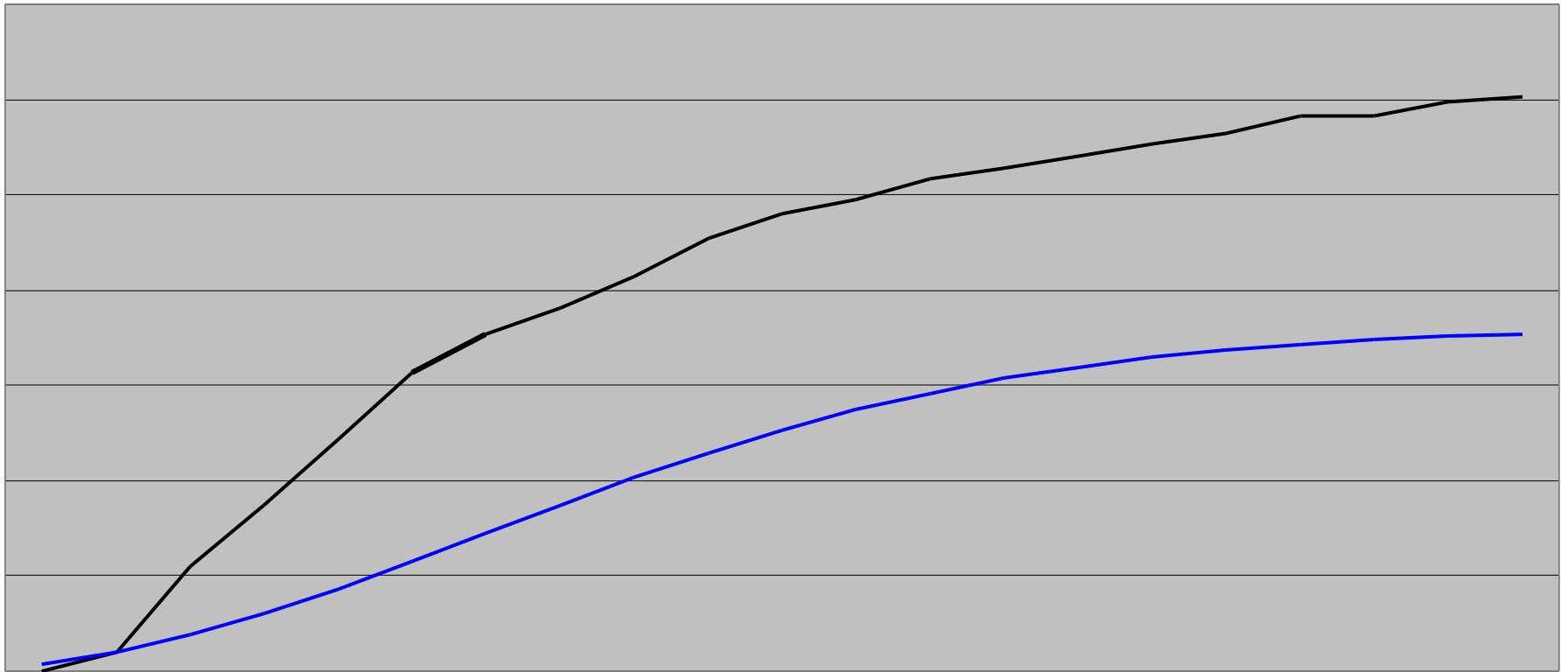
Cumulative Bugs closed

Does this bug arrival chart tell you if you are on schedule?

What does it tell you about your staffing level?

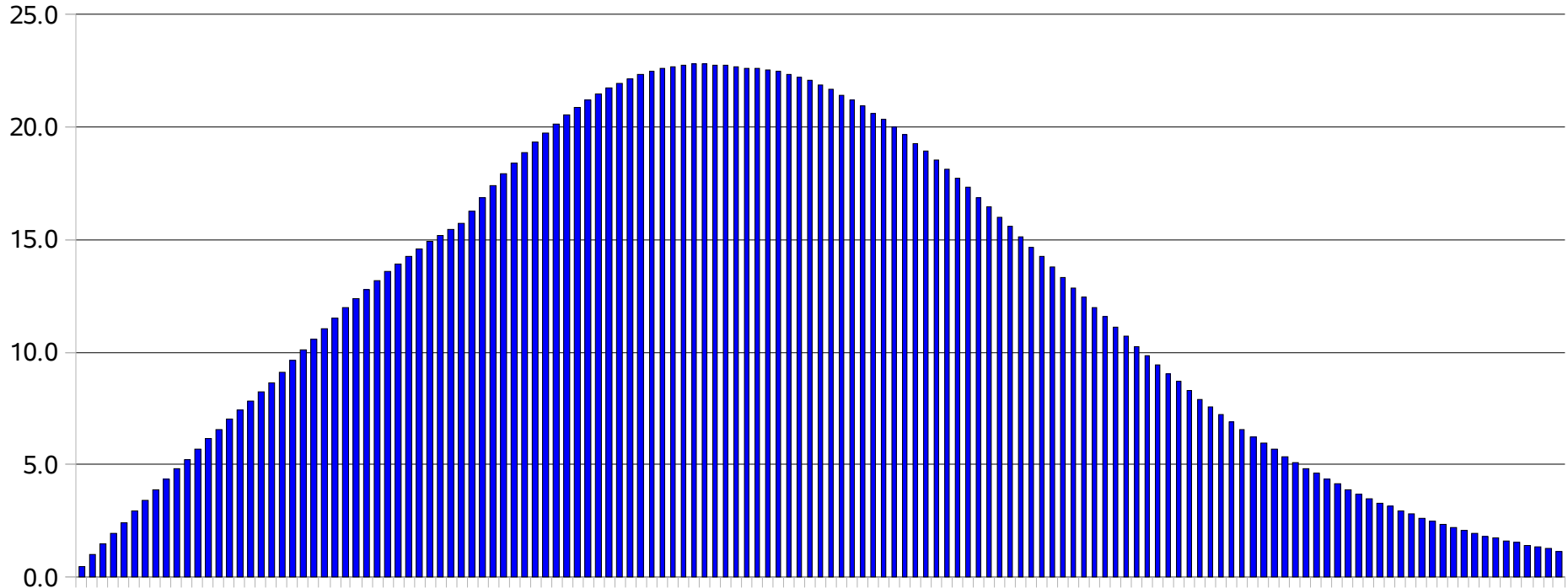


Something is Wrong Here!

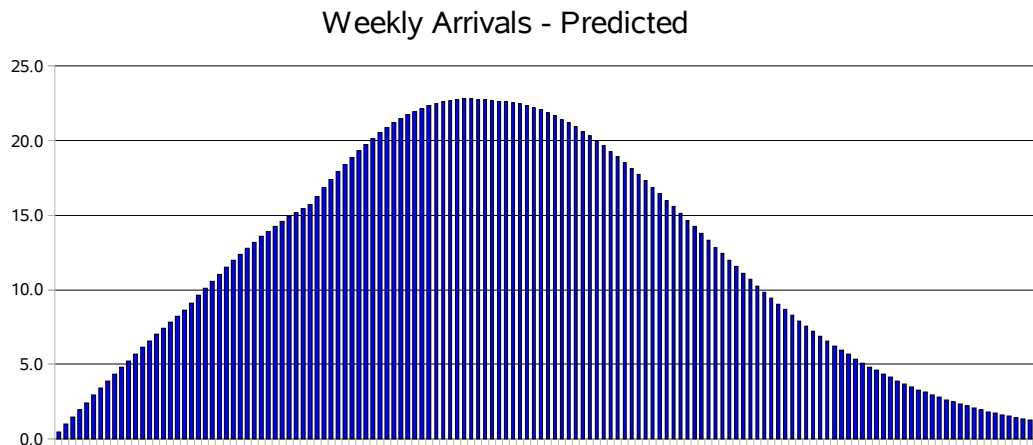


Using Projections for Planning

Weekly Arrivals - Predicted

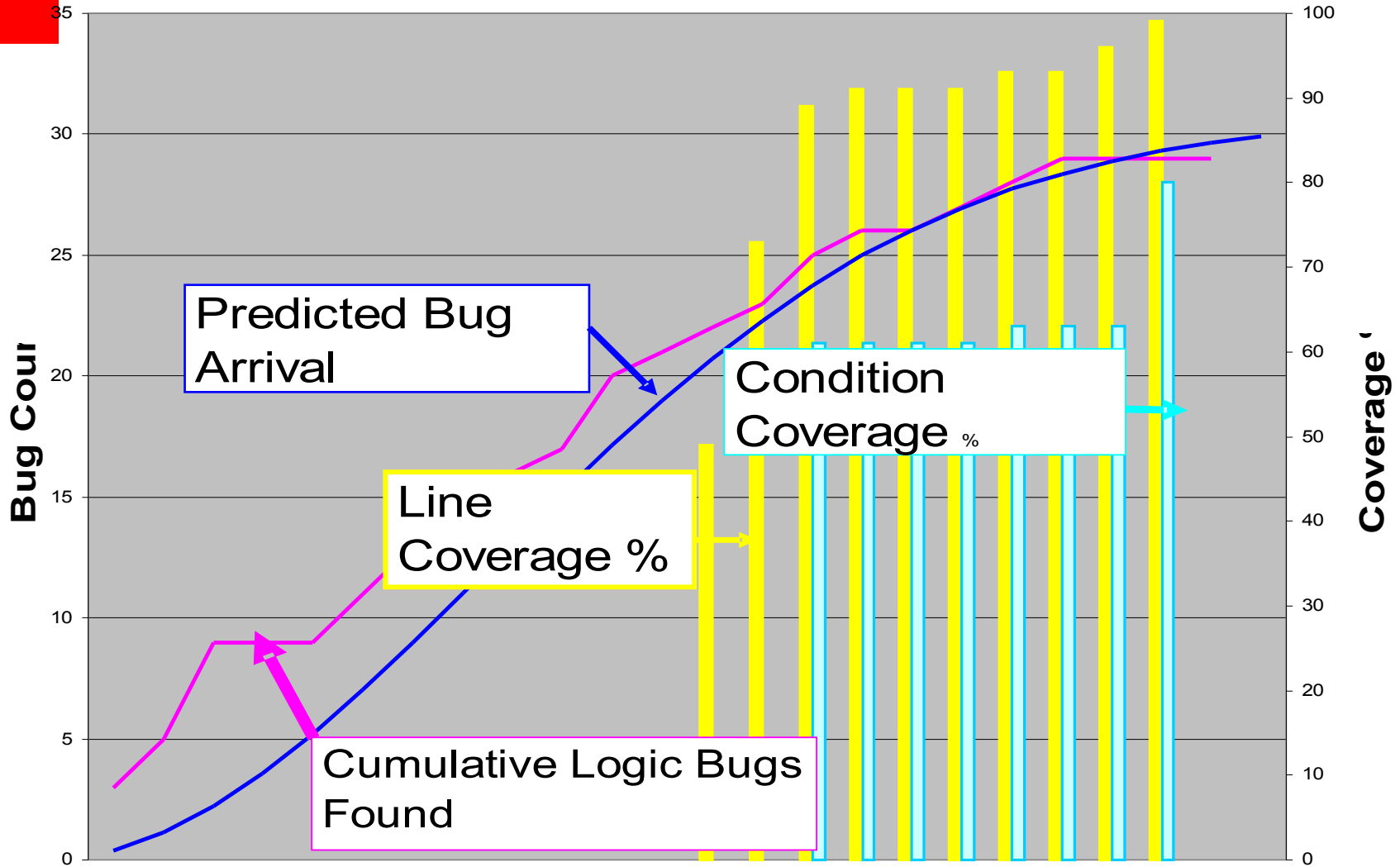


Using Projections for Planning



- Can tell you how many resources you need
 - To find max # bugs/week = how many resources?
- Can help you plan release points
 - Go to emulation when 75% of bugs are found
 - Start freeze process when 90% of bugs are found

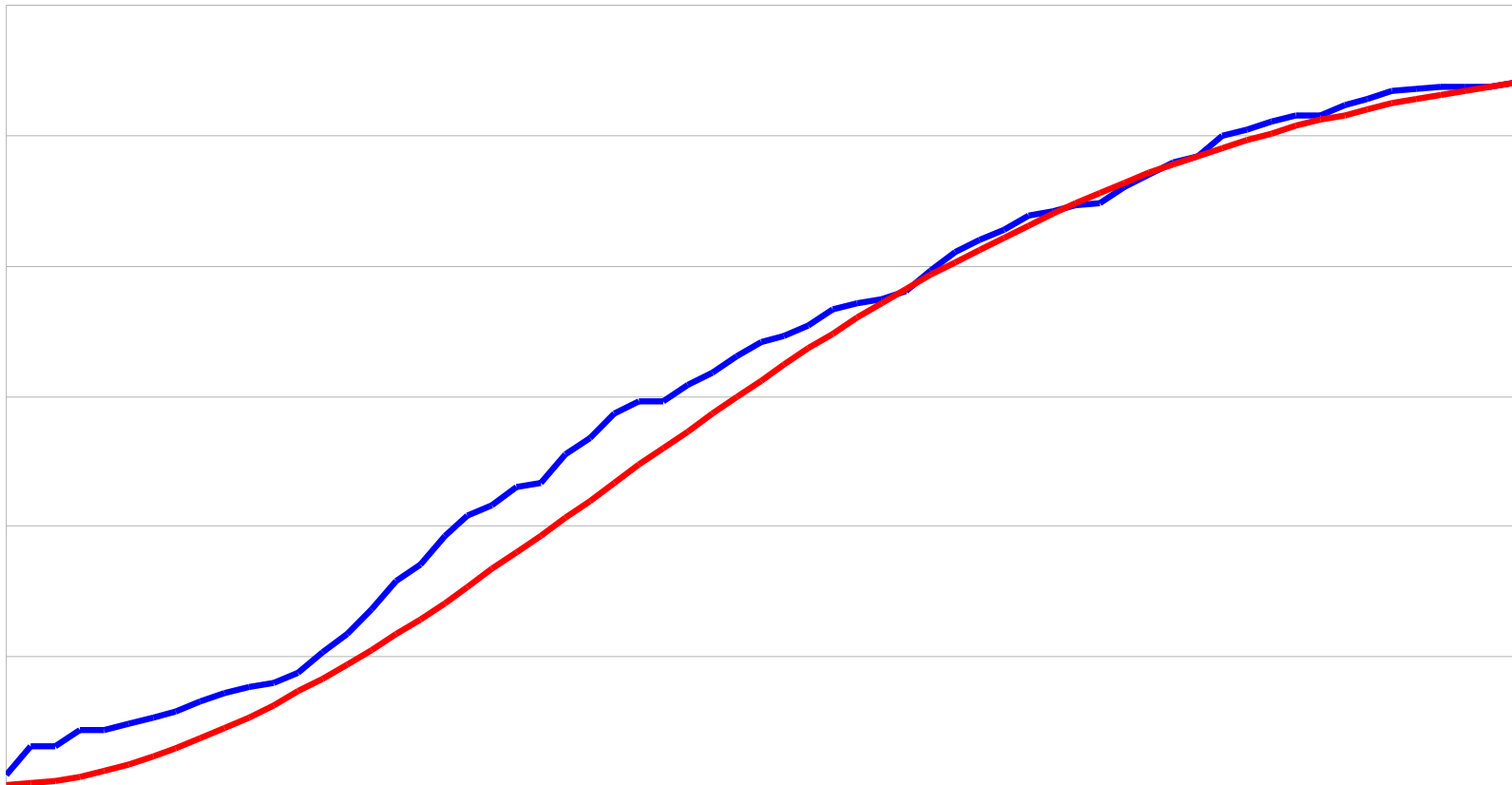
Chip Verif Progress "at a glance"



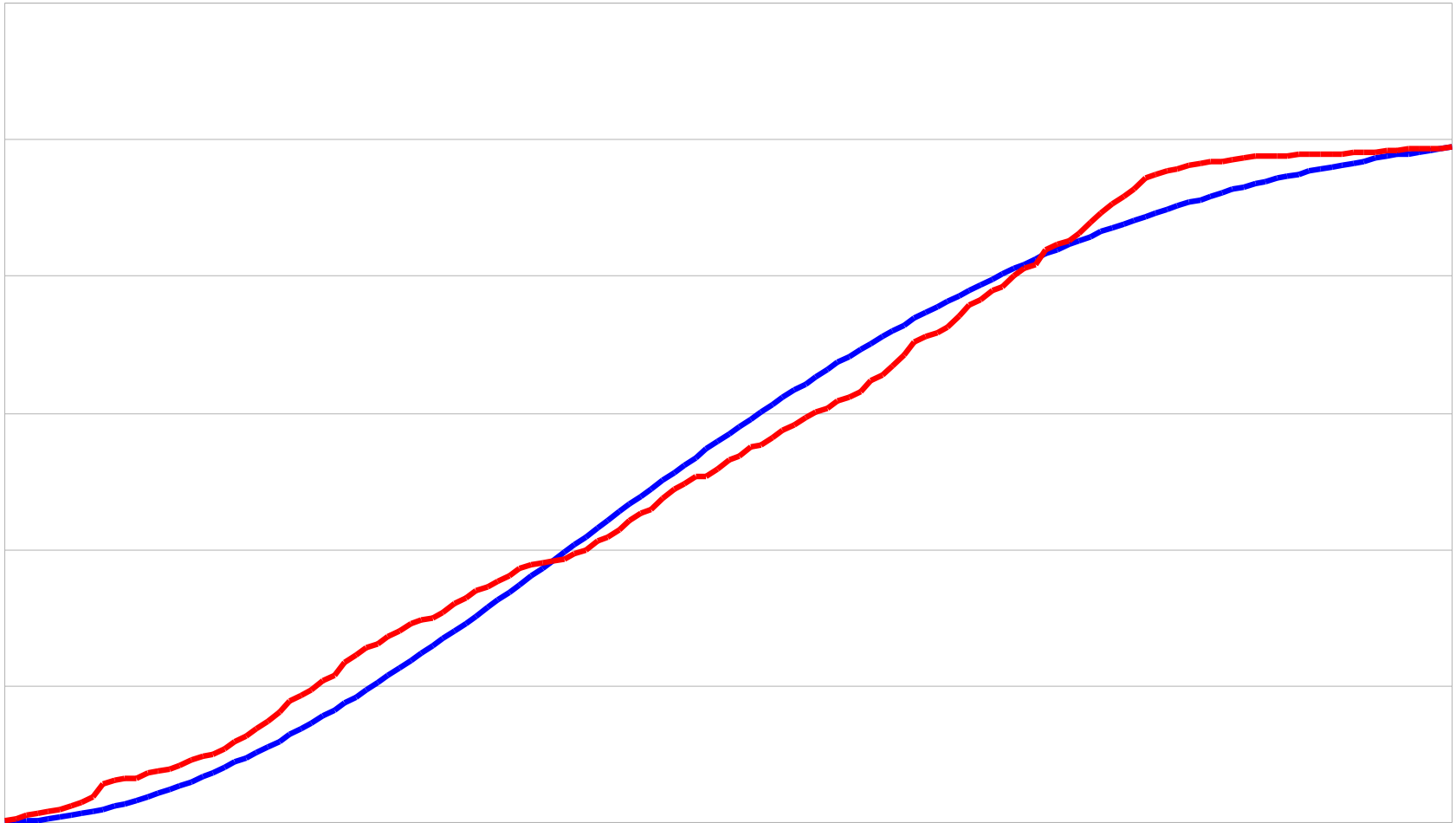
Is this a one trick pony?

- HP Results are for ASICs & large FPGA
 - Taped out 10 bug free chips that followed the model to a high degree
- Robert Page applying this technique at Freescale
 - Kim Asai (TI) and Rahman Farhan (AMD) expressed interest
- What about processor projects?
- Researched past projects at Oracle to see:
 - > Does the arrival rate equation match?
 - > What can I use to calculate bug density?
- Applied to a recently completed processor project with success

SOC Blocks Arrivals vs Predicted



Ultra SPARC T2 Core Actual vs Predicted Arrivals

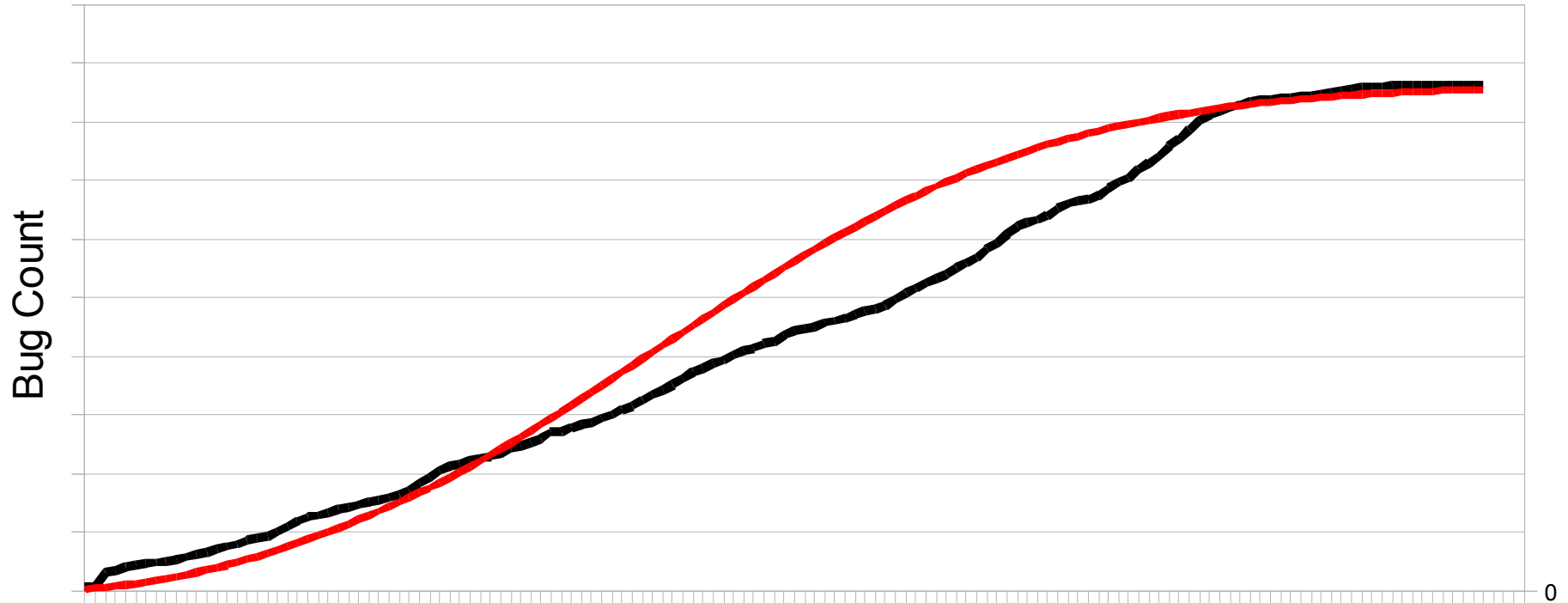


Bug Density in Processor Projects

- Cant use bugs / Line. Structural coding style makes using lines of code problematic
- What else measures design size?
- Code coverage metrics gave me an idea: # of Conditions, as reported by condition coverage metric might be usable
- Discovered decent correlation amongst projects

Actual chart from recent processor

Actual Bug Arrivals vs Predicted



- Predicted bug count was off by less than 1%

How can I use this: Determining your bug density factor.

Bugs per ???

- If you have Behavioral Verilog – use LOC
- Applying to a different design/coding style will require work
- Research past projects and look at historical bug density
- Bottom line – you need to do some homework but you can find something that you can work with

Conclusion

- How to answer that difficult question?
- In the SW world, I found a technique I was looking for:
 - > **Quantitative**– calculated value
 - > **Predictive** – am I on track? When might I be done?
- Rayleigh arrival model borrowed from software
- To use this technique you need to:
 - > Assess your own RTL team's defect density - Use past project's historical data

Conclusion

- You can create a viable method that can be used to track your bug discovery rate vs. an expected rate
 - This provides a point of reference that you can use to assess the state of your project.
- Combined with traditional metrics, the technique can contribute to your assessment of “Am I Done?”, but also help answer the question:

When will you be done?

Thank you for your attention.

ORACLE®