Fundamentals of High-Level Synthesis — Part 4: Dependency, Concurrency and parallelism

Mohammad Hosseinabady · Following

4 min read · Nov 18, 2019

Image by Ranya from Pixabay

In the last two blogs, I explicitly talked about concurrency and parallelism. I also mentioned about dependency. However, the dependency concept and its relation to concurrency and parallelism need more discussion. Here, I am going to talk about this relationship.

Maximising parallelism is our ultimate goal in accelerating an algorithm using FPGA. This goal is achievable, mainly due to the abundant computational and memory resources available. However, utilising parallelism is not easily accessible if enough concurrency doesn't exist in the task description. As I mentioned in the previous post, one way of describing the concurrency in the code is using proper programming patterns such as *map*. However, it is not, all the time, easy to find a suitable programming pattern. In this case, the concept of dependency can be helpful. In addition, the concept of dependency can help to find the proper concurrency pattern.

Dependency is a technique that usually used by compilers to exploit concurrency. Dependency analysis is a formal theory that investigates the concurrency and order of execution among statements in a program. Directed graphs are conventional approaches to model dependencies in a program. The node in this graph represents instructions, and the links denote the dependency among instructions.

There are two types of dependencies in a piece of code: *data dependency* and *control dependency*.

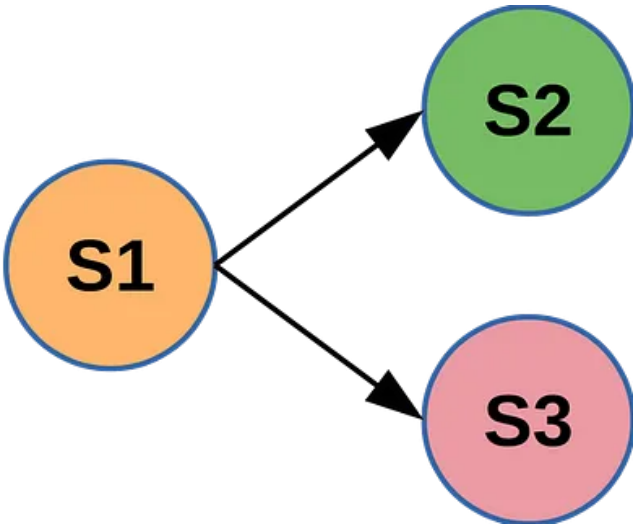
Two instructions are data-dependent if the execution of one requires the results of the other.

A data dependency graph (or dataflow graph) can show this relationship. The following figure shows the relation between two instructions *S1* and *S2*. In this graph, the S2 instruction (or destination node) requires some data from its predecessor instruction, i.e., S1 (or source node).

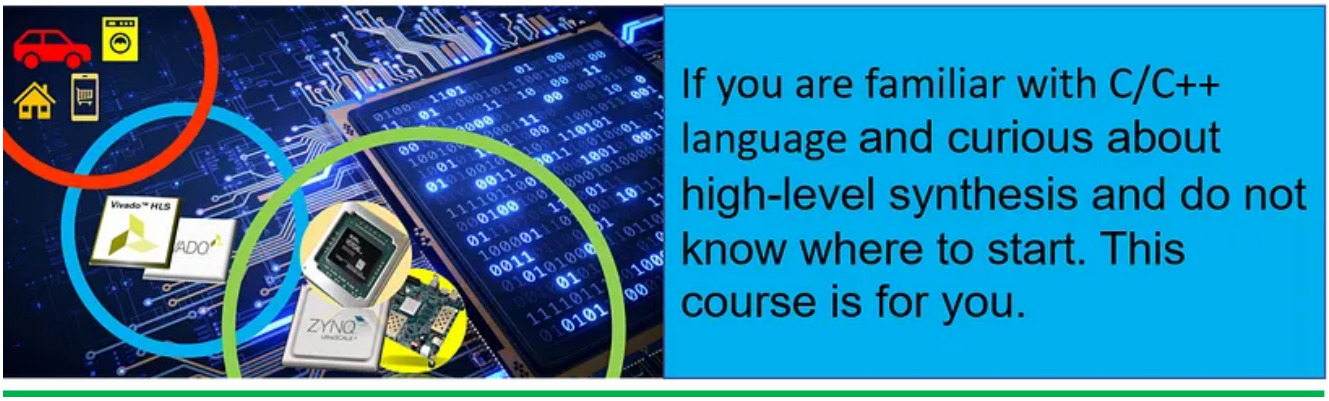
Note that, in a dataflow graph, execution of the source node results in the execution of the destination node and there is no way to stop destination node execution. This is in contrast to the control dependency.

In control dependency, the execution of a node depends on the results of another node. For example, there is a control dependency between *S1*, *S2* and *S3*, as the *S1* instruction controls the execution of *S2* and *S3*.

```
S1: if (a > 2)
S2:    x = y + 2;
     else
S3:    x = z + 2;
```



Dependencies among instructions or code blocks are more critical when they are combined with iterative structures like *for*-loops. In an iterative structure, not only the dependency between instructions inside an iteration is important but also the dependency between instruction in two or more adjacent iterations needs special attention.



But, how can we use dependency analysis to develop a better design in HLS? I address this question with an example.

Understanding and reducing dependencies among instructions and code blocks can help programmers to develop more efficient code. Let us consider the *sgemv* operator from BLAS. This operator is a matrix-vector multiplication shown by the following equation, where *a* and *b* are two float constants, *x* and *y* are two vectors, and is a matrix of size *NxM*.

The following code shows an implementation of this operator in Xilinx Vivado-HLS. It consists of two nested *for*-loop.

```
#define M 32
#define N 32
void sgemv_accel(float A[N][M], float x[M], float y[N],
  float alpha, float beta)
{
#pragma HLS ARRAY_PARTITION variable=A complete dim=1
#pragma HLS ARRAY_PARTITION variable=A complete dim=2
#pragma HLS ARRAY_PARTITION variable=x complete dim=1
#pragma HLS ARRAY_PARTITION variable=y complete dim=1

  for (int i = 0; i < N; i++) {
    y[i] = beta * y[i];
    for (int j = 0; j < M; j++) {
#pragma HLS UNROLL
      y[i] += alpha*x[j]*A[i][j];
    }
  }
}
```

The inner loop is unrolled for parallel execution. However, the data dependency between two adjacent iterations prevents parallelism. After synthesising this function, considering the ZynqMPSoC-based Ultra96v2 Board, the latency of the inner loop is 134 clock cycles, and the latency of the entire function is 4289 clock cycles.

It is possible to reduce the data dependency by using the loop interchange transformation. The following code shows the transformed code. The first loop initialises the *y* vector with the second term in the *sgemv* equation. The second nested loops perform the first term in the *sgemv* equation.

```
#define M 32
#define N 32
void sgemv_accel(float A[N][M], float x[M], float y[N],
                 float alpha, float beta) {
#pragma HLS ARRAY_PARTITION variable=A complete dim=1
#pragma HLS ARRAY_PARTITION variable=A complete dim=2
#pragma HLS ARRAY_PARTITION variable=x complete dim=1
#pragma HLS ARRAY_PARTITION variable=y complete dim=1

  for (int i = 0; i < N; i++) {
#pragma HLS UNROLL
    y[i] = beta * y[i];
  }
  for (int j = 0; j < M; j++) {
    for (int i = 0; i < N; i++) {
#pragma HLS UNROLL
      y[i] += alpha*x[j]*A[i][j];
```

```
      }
    }
  }
```

As there is no data dependency between adjacent iterations of the inner-loop, it can be parallelised by unrolling the loop. After synthesising this code with Xilinx Vivado-HLS and considerin the ZynqMPSoC-based Ultra96v2 Board, the latency of the inner loop would be 11 clock cycles, and the total latency is 356 clock cycles, which is 4289/356=12.05 times faster the first implementation.

In summary, the result of every dependency in a code is poor parallelism; analyse the code, find the dependencies and resolve them for more performance.

. . .

Originally published at http://highlevel-synthesis.com on November 18, 2019.

Programming | Software Development | Accelerator | Hardware | High Level Synthesis

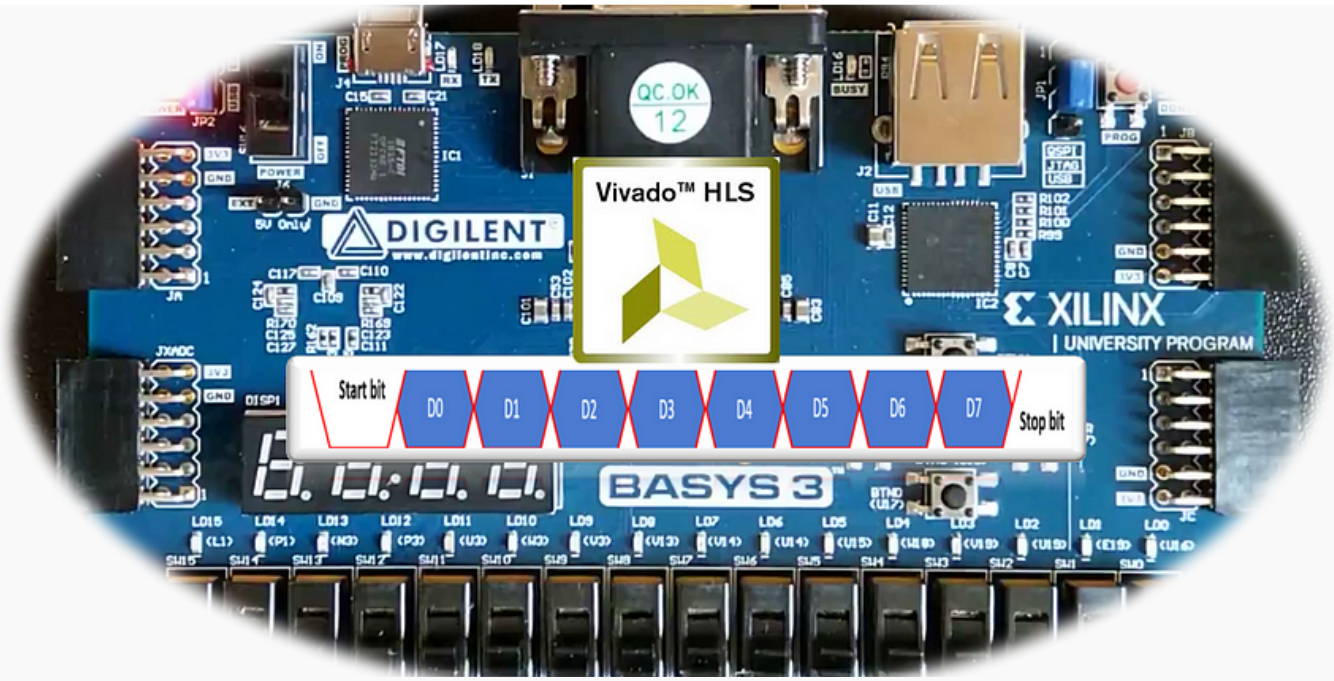
Following

Written by Mohammad Hosseinabady

39 Followers

Designing digital systems and accelerating functions with HLS for FPGAs are fun.

More from Mohammad Hosseinabady



Mohammad Hosseinabady

UART Transmit with HLS for FPGA

Like my previous projects, this one also demonstrates that “Designing digital systems with HLS for FPGA is fun”. If you are interested in...

3 min read · Oct 26, 2020

Mohammad Hosseinabady

“Logic System Design with High-Level Synthesis": an Introductory Course for Undergraduate

This article introduces an introductory course on high-level synthetisis (HLS) that is suitable for undergraduate students and software or...

· 3 min read · Apr 15, 2021

9

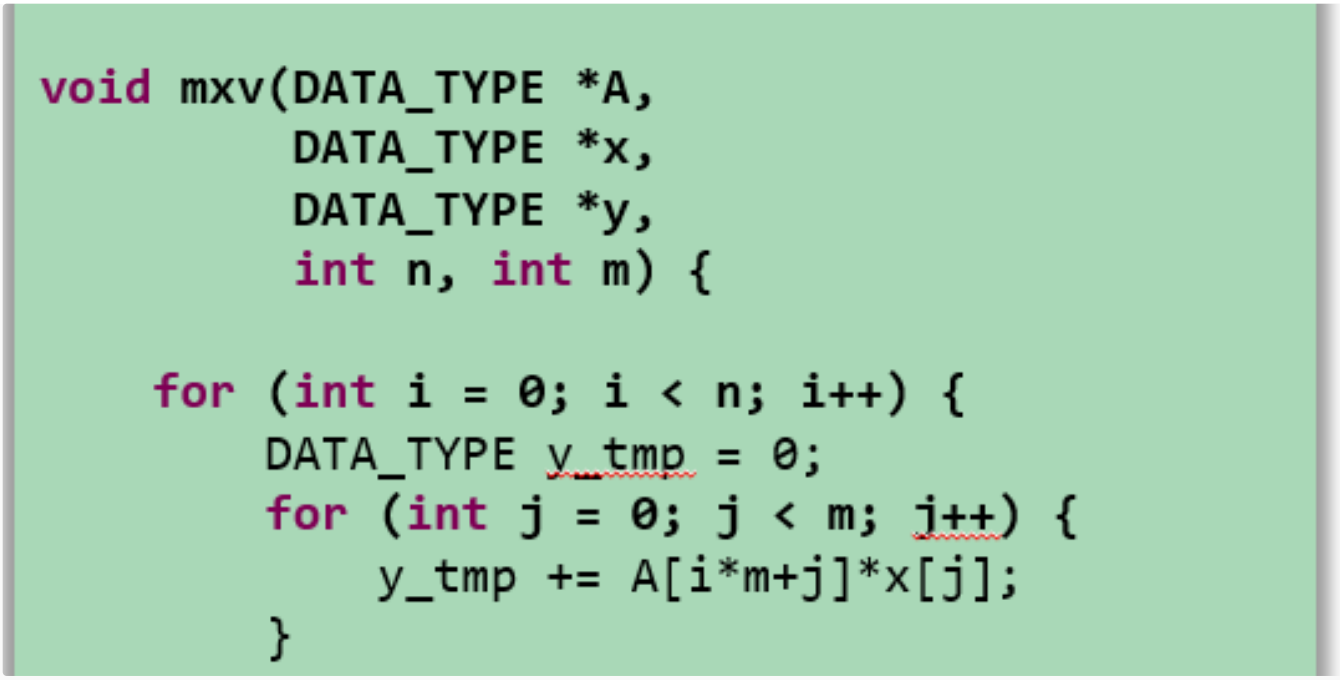
Mohammad Hosseinabady

Vitis 2021.1 Embedded Platform for Zybo-Z7–20

The goal of this blog is to create a Vitis 2021.1 hardware accelerator platform for the Zybo-Z7–20 board from Digilent.

· 8 min read · Aug 16, 2021

5



```
void mxv(DATA_TYPE *A,
         DATA_TYPE *x,
         DATA_TYPE *y,
         int n, int m) {

    for (int i = 0; i < n; i++) {
        DATA_TYPE y_tmp = 0;
        for (int j = 0; j < m; j++) {
            y_tmp += A[i*m+j]*x[j];
        }
```

Mohammad Hosseinabady

How to Reduce II in HLS: Part 4

This week's problem is the traditional matrix-vector multiplication kernel used in several applications such as machine learning and image...

· 1 min read · May 24, 2021

See all from Mohammad Hosseinabady

Recommended from Medium

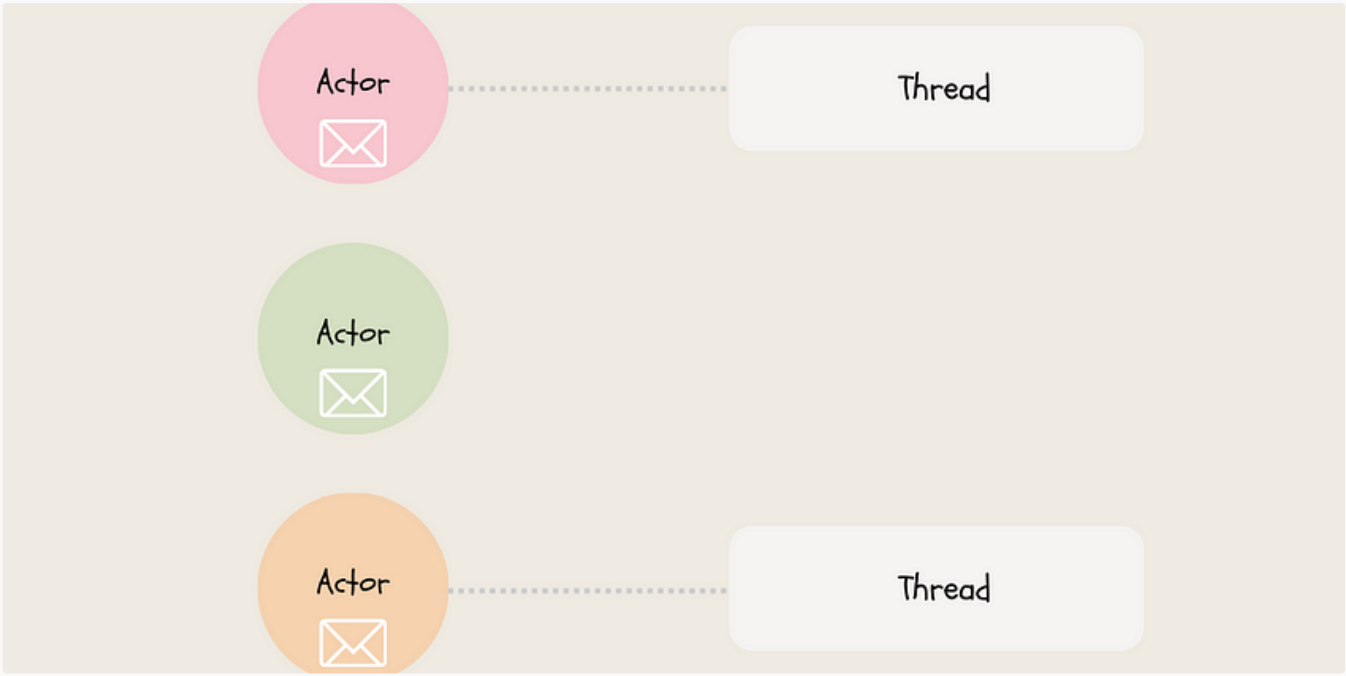
Vaishnav Manoj in DataX Journal

JSON is incredibly slow: Here's What's Faster!

Unlocking the Need for Speed: Optimizing JSON Performance for Lightning-Fast Apps and Finding Alternatives to it!

16 min read · Sep 28, 2023

12K 142



Nidhey Indurkar

How did PayPal handle a billion daily transactions with eight virtual machines?

I recently came across a reddit post that caught my attention: ‘How PayPal Scaled to Billions of Transactions Daily Using Just 8VMs’...

7 min read · Jan 1

2.1K 25

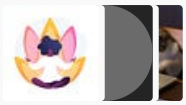
Lists

General Coding Knowledge

20 stories · 802 saves

Coding & Development

11 stories · 387 saves



Stories to Help You Grow as a Software Developer

19 stories · 722 saves

Leadership

41 stories · 201 saves

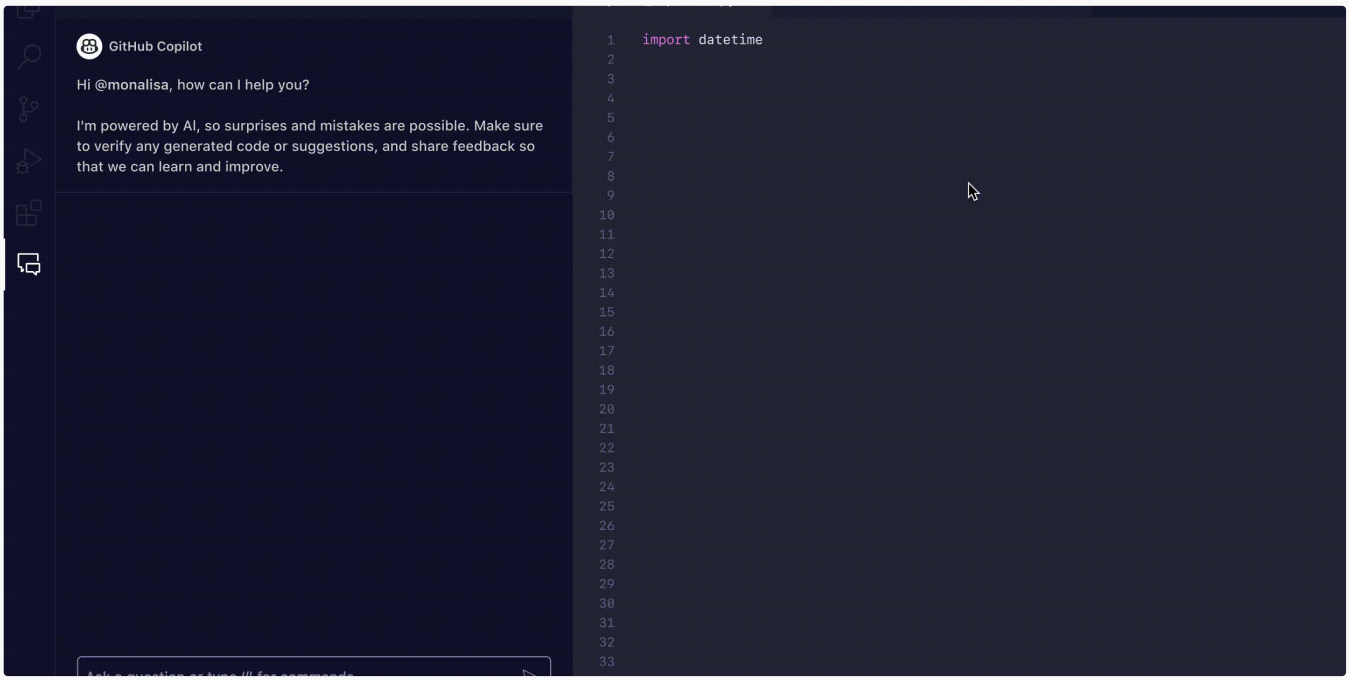
Benoit Ruiz in Better Programming

Advice From a Software Engineer With 8 Years of Experience

Practical tips for those who want to advance in their careers

22 min read · Mar 21, 2023

13.5K 252



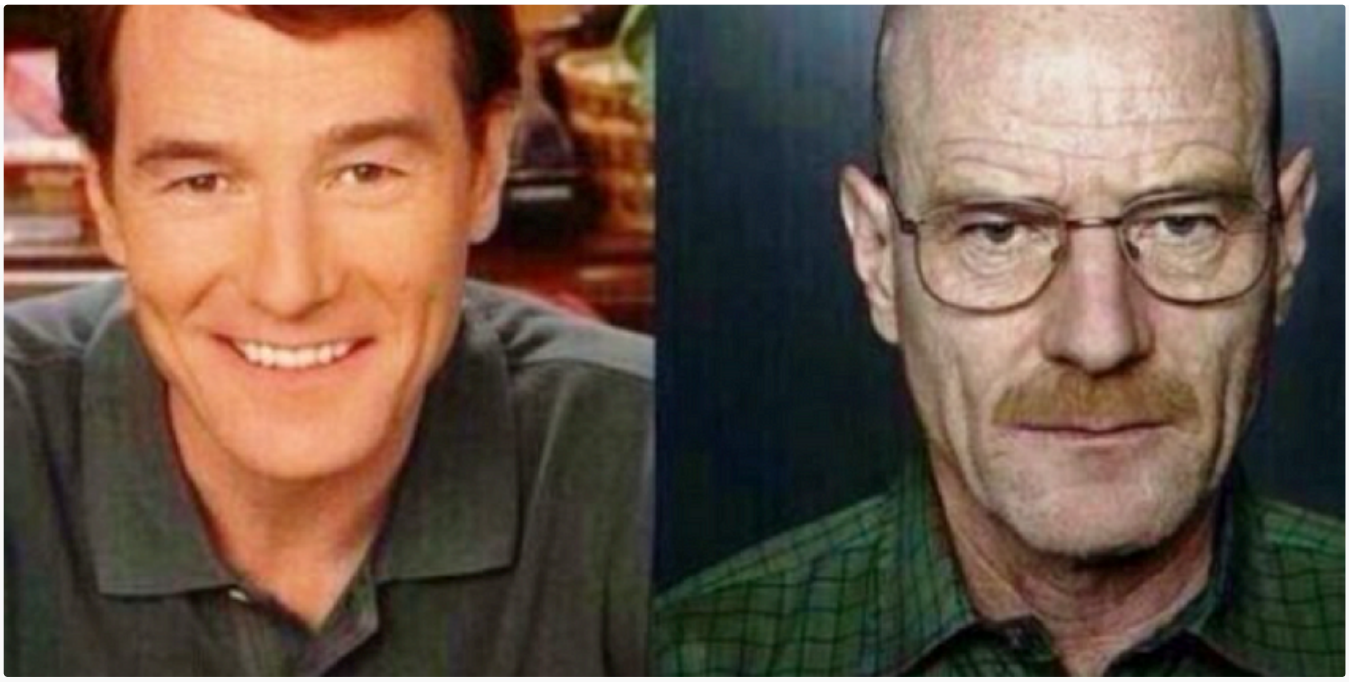
Jacob Bennett in Level Up Coding

The 5 paid subscriptions I actually use in 2024 as a software engineer

Tools I use that are cheaper than Netflix

· 5 min read · Jan 4

2.9K 42



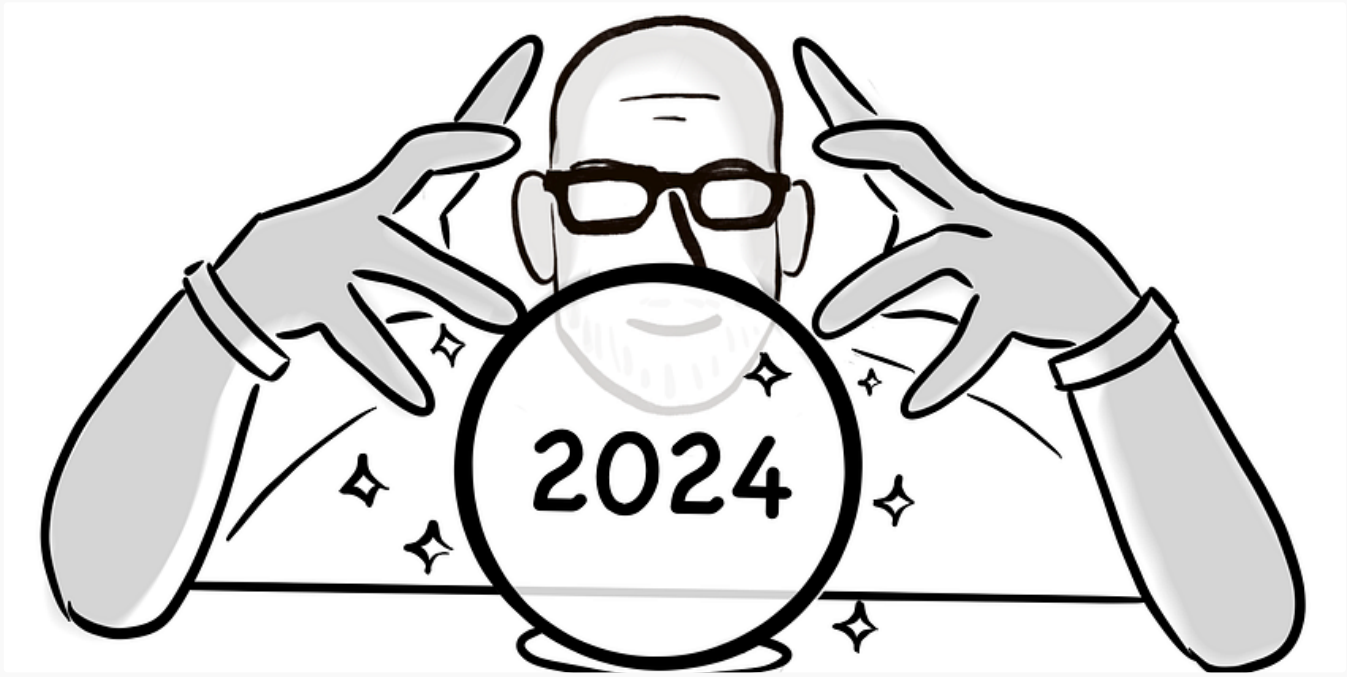
David Goudet

This is Why I Didn't Accept You as a Senior Software Engineer

An Alarming Trend in The Software Industry

5 min read · Jul 26, 2023

7.7K 80



Scott Galloway

2024 Predictions

Each year, we review/make predictions re the past/coming year. Most years, we hit more than we miss. But we do miss—if we made 10...

11 min read · Jan 6

9.4K 123

See more recommendations