Confirming the Design Gap

Benjamin Menhorn and Frank Slomka

Institute for Embedded Systems/Real-Time Systems
Ulm University
Albert-Einstein-Allee 11, Ulm, Germany
benjamin.menhorn|frank.slomka@uni-ulm.de

Abstract. The design gap is the divergence between technology capabilities and design capabilities. This work uses a complexity measure to calculate complexity figures for highly regular and irregular structures. This measurement allows to assess complexity without the need of empirical data to chart the design gap. This will demonstrate that the design gap is caused by the structure itself.

1 Introduction

The design gap is well-known to digital hardware designers and others. It shows the spread between technology capabilities and hardware design capabilities. Among others, one explanation lists the structure of designs as cause for the design gap. For now, there are only reasonable explanations for the structure as source. The aim of this work is to show, that the design gap is caused by the structure itself. Therefore a complexity measurement method is applied to designs with different degrees of regularity.

This approach faces two main challenges. On the one hand two different designs have to reflect the technology capabilities and hardware design capabilities. On the other hand an adequate measurement method for complexity has to be found. The complexity measure has to provide figures as mathematical statements about the designs.

This work is organized as follows: First, the design gap is discussed with its possible explanations in related work. Then, it will be discussed why a memory as a highly regular structure and a processor as a highly irregular structure can reflect technology capabilities and hardware design capabilities. The next part introduces a hardware design measurement which is used to calculate complexity. With all presuppositions the last part introduces and analyzes the designs and reveals the design gap. The work closes with the conclusion.

2 Related Work

2.1 Moore's Law

In context with integrated circuits, Moore's Law describes the long-term trend for the amount of transistors which can be placed on a chip inexpensively. Depending on the source, the amount doubles every 18 to 24 months [1] [2] [3].

Figure 1 charts the trend of integrating transistors on a single chip over the past two decades. Even though Moore's Law is not a scientific natural law it is widely accepted as an observation prediction of integrated circuit development. At the same time one can speak of a "self-fulfilling prophecy" since various industries are involved in the development of better microchips [4].

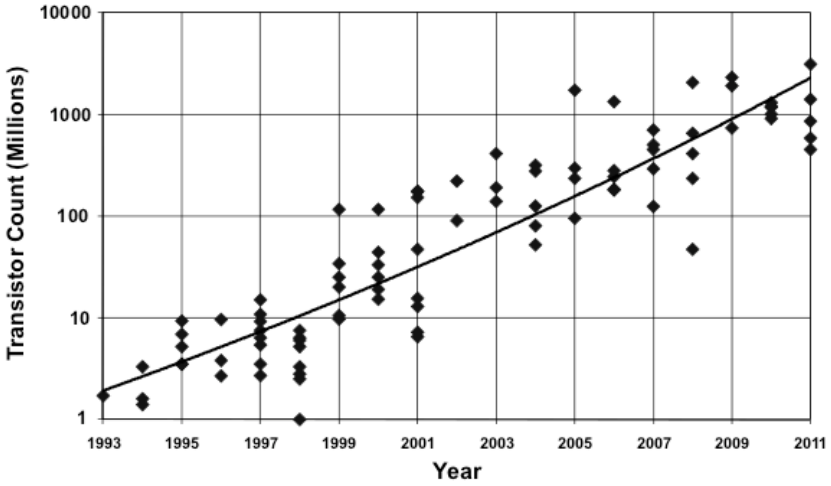

Fig. 1. Chip complexity development (from [5])

Originally Gordon Moore observed the number of components per integrated function [1]. Later, transistors per chip were counted instead of components. Even though Moore's Law counts transistors over time, it is often used to describe the development of complexity over time [6]. The figures are based on observations and forecasts. In order to compare the complexity from Moore's Law with the productivity, the following section will introduce the design gap.

2.2 The design gap

The design gap is the divergence between technology and design capabilities. Design capabilities (also called productivity) are measured in transistors per day [7]. Technology capabilities (also called complexity) are measured in transistors per chip as in Moore's Law [7]. **Figure 2** plots productivity and complexity over time. The design gap can be clearly identified. Complexity has an annual growth rate of 58% while productivity has an annual growth rate of 21% [8]. A common opinion for the slower growing productivity found in literature bases the design gap on missing tools, abstraction layers and design possibilities [9] [10]. Another work states, that the "designers appear limited by the very tools and processes that made those billion transistors a reality" [11]. As well as that the "design gap

might be avoided if more ESL (**E**lectronic **S**ystem **L**evel) methodologies would be deployed" [12].

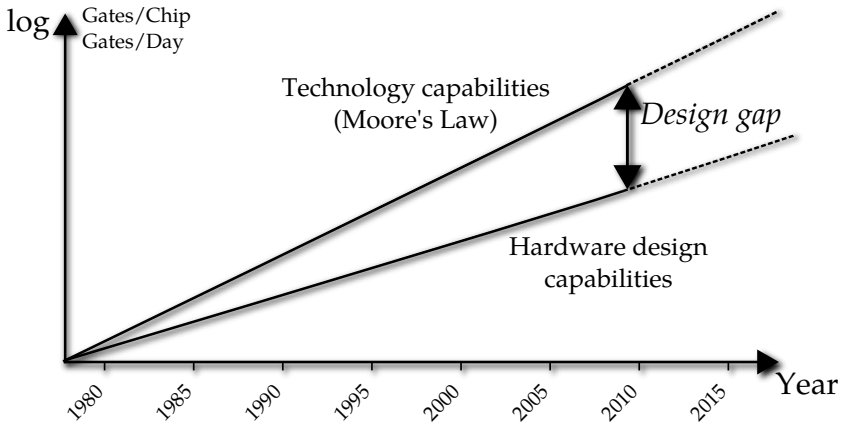

Fig. 2. Design gap (Data from [13])

This works advances the view that the design gap is originally caused by the difference between regular and irregular structures itself. Therefore the following section will identify designs representing highly regular and highly irregular structures as well as introduce a complexity measure.

3 Presuppositions

In order to calculated complexity for charting the design gap, two challenges appear. The first challenge is to identify a design which represents the technology capabilities (Moore's Law) and another design representing the design capabilities (productivity). The second challenge is to find an adequate complexity measure for hardware designs.

3.1 Structure regularity

The design gap is the disparity between transistors available to a designer and the ability to use them effectively in a design [9]. The technology capabilities in Moore's Law represent the amount of transistors which can be placed on a chip inexpensively. The highest gate density can be found in memories, a highly regular structure. Breaking it down to the basic components, a memory consists of single memory cells and an inverse multiplexer as control unit. With the development of one (elementary) memory cell the basic component for a memory is designed. This cell can be copied in any quantity and put together to a grid. This grid and a separately developed control logic compose a whole

memory. The reusability of single memory cells and the grid arrangement reduce the design complexity drastically.

On the other hand, a design representing the productivity can be found in a processor. Processors are highly irregular structures. Their control logic needs most of their chip area. This is a highly irregular structure which is more complicated to design. There are no elementary cells which can be copied in any quantity. But parts of the processor such as caches or registers also consist of single, reusable components. Therefore processors are not completely irregular structures.

In contrast to memories, reusability in processors is very limited. This increases design complexity. By adding regular structures, for example by increasing the processor's memory (cache), the throughput can be increased. By reducing the complexity of irregular structures at the same time, e.g. by using RISC architectures, the ratio between regular and irregular structures can be enhanced in favor of less complex designs.

The design gap is caused by this difference in structure. Additional abstraction layers and design tools shorten the gap between technology capabilities and design capabilities. Those approaches reduce the decision possibilities for designers and engineers. Thereby the amount of possible states of a system is reduced. This is especially interesting in design automation processes.

3.2 Complexity measure

The determination of complexity makes it possible to give system related statements and allows to compare different systems [14] [15]. In order to develop a complexity model, complexity itself needs to be understood [16]. But today, most methods for estimating system size use empirical data by analyzing previous projects [17]. In [18] a measurement, called design entropy concept, was proposed which doesn't rely on empirical data. This concept bases its calculations on an abstract variable: states. "A state is a situation in which a system or system's component may be at a certain point of time. It refers to the interior of a system and ignores external influences such as input and output. The set of states is the abstraction of a real system" [19].

The main statements of the design entropy concept are summarized in the following in order to calculate complexity for the different designs in the next section. The approach of the design entropy bases on Shannon's information entropy. In order to give mathematical statements about transmitted information, Claude Elwood Shannon developed a model which became famous as Shannon's information theory [20]. The design entropy concept identifies the single components of a design as sources and drains of information. Connections between components are channels and information are symbols transmitted from a pool of available symbols. In digital hardware connections are normally implemented by wires. The available symbols are "high" and "low" signals in the simplest model. For instance an assignment `a:=b` between two components would be identified as: The information *(=signal level)* from component *(=source/sender)* `b` is transmitted *(=assigned)* to component *(=drain/receiver)* `a`.

Complexity can be considered to be a measure of a system's disorder which is a property of a system's state. Complexity varies with changes made at the amount of possible states of a system. An entropy measurement can be used to measure project size by using states. States are abstract variables which depend on the analyzed property of a project. It becomes possible to calculate the effect of introducing design tools to a development process by calculating complexity on different abstraction levels and comparing them.

$$H = -K \sum_{i=1}^{N} p_\alpha \log p_\alpha \tag{1}$$

The form of Shannon's theorem (see equation (1)) is recognized as that of entropy as defined in certain formulations of statistical mechanics (e.g. [21]), where p_α is the probability of a system being in cell α of its phase space. H is from Boltzmann's famous H theorem and the constant K merely amounts to a choice of unit of measure [20]. Equation (1) can be rewritten as (2) and leads to definition 1 [18] [22].

Definition 1 (Behavior Entropy) *Let c be a component with inputs (component's sources)* $n_i(c)$, $i = \{1, \ldots, n(c)\}$ *and outputs (component's drains)* $m_j(c)$, $j = \{1, \ldots, m(c)\}$, *where* $n(c)$ *is the amount of inputs of c and* $m(c)$ *the amount of outputs of c. Let* $z(n_i(c))$, $i = \{1, \ldots, n(c)\}$ *be the amount of possible states for the inputs* $n_1(c) \ldots n_{n(c)}(c)$ *and* $z(m_j(c))$, $j = \{1, \ldots, m(c)\}$ *be the amounts of possible states for the outputs* $m_1(c) \ldots m_{m(c)}(c)$. *Then the behavior entropy* $H_B(c) \in \mathbb{R}_0^+$ *of component c is given by:*

$$H_B(c) = log \left(\prod_{i=1}^{n(c)} z(n_i(c)) \cdot \prod_{j=1}^{m(c)} z(m_j(c)) \right) \tag{2}$$

The behavior entropy gives a statement about the (usage) complexity of a component. The behavior complexity does not allow for the actual implementation complexity of a component. It only provides complexity information about the usage of a component. It can be compared to an outer or black box view on a component. In contrast to the behavior entropy the structure entropy in definition 2 provides information how complex the implementation/realization of a component is. This is similar to an inner or white box view on a component.

Definition 2 (Structure Entropy) *Let c be a component with instances* c_b *and implemented sub-components* c_s. *Then the structure entropy* $H_S(c) \in \mathbb{R}_0^+$ *for component c is given by the sum of all behavior entropies of all instances* c_b *and the structure entropy of all implemented sub-components* c_s*:*

$$H_S(c) = \sum_{i \in c_b} H_B(i) + \sum_{j \in c_s} H_S(j) \tag{3}$$

The structure entropy allows to add up the entropies from the single sub-components. If these sub-components have also been implemented, their structure complexity needs to be considered, too.

In the following section the measurement will be applied on a well-known effect, the design gap. This will demonstrate on the one hand, that the source for complexity lays in the structure itself and on the other hand that the design entropy model is capable to give mathematical statements about the complexity of designs.

4 Explaining the design gap

In order to demonstrate that the design gap is caused by the structure itself, we implemented a memory and a processor in VHDL. For both implementations, only basic gates (such as AND, OR, XOR, NOR etc.) were used in order to make both designs comparable. In order to demonstrate the design gap, we need to calculate the complexity. Because both implementations consist of several single components, we will show the complexity calculation with a memory cell. Therefore, the structure of the memory will be described in the following and the VHDL code will be given.

4.1 Memory

As discussed before, this work considers a memory as a design where transistors can be placed on a chip inexpensively. The memory is linked to Moore's Law and the technology capabilities. The fundamental structure of a memory in shown in **figure 3**. The main parts are an address decoder and the storage matrix. The storage matrix consists of single memory cells. The structure of a memory cell in shown in **figure 4**. The core is a D-flip-flop, which can hold one bit. The VHDL implementation of such a memory cell is given in **listing 1.1**.

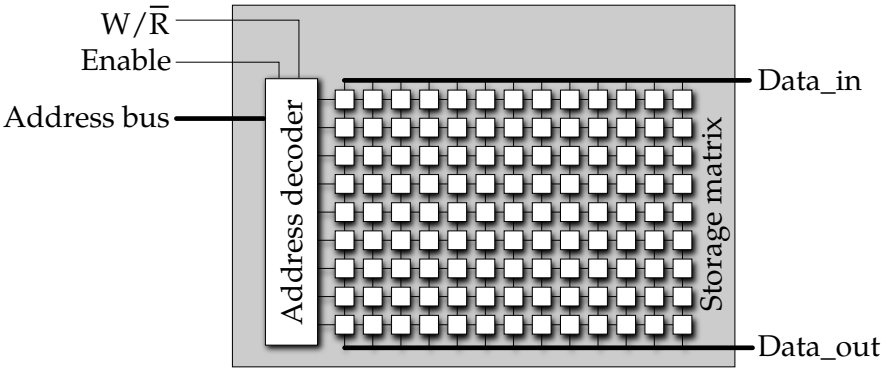

Fig. 3. Fundamental structure of a memory

```
1  entity memory_cell is
2  port (
3       I           :in  std_logic; --input bit
4       W           :in  std_logic; --write_enable
5       S           :in  std_logic; --select_enable
6       O           :out std_logic  --output bit
7  );
8  end memory_cell;
9
10 architecture structural of memory_cell is
11      signal E,D_g,E_g,Qa,Qb,NotD :std_logic;
12
13 begin
14      E <= W AND S;
15      NotD <= Not I;
16      D_g <= NotD AND E;
17      E_g <= I AND E;
18      Qa <= Qb NOR D_g;
19      Qb <= Qa NOR E_g;
20      O <= Qa and S and not W;
21 end structural;
```

Listing 1.1. VDHL code for a memory cell

In order to calculate the complexity for the memory equations (2) and (3) are used. The calculation starts with the complexity of a single memory cell:

$$H_B(\text{memory\_cell}) = 4 \cdot \log(2) \tag{4}$$

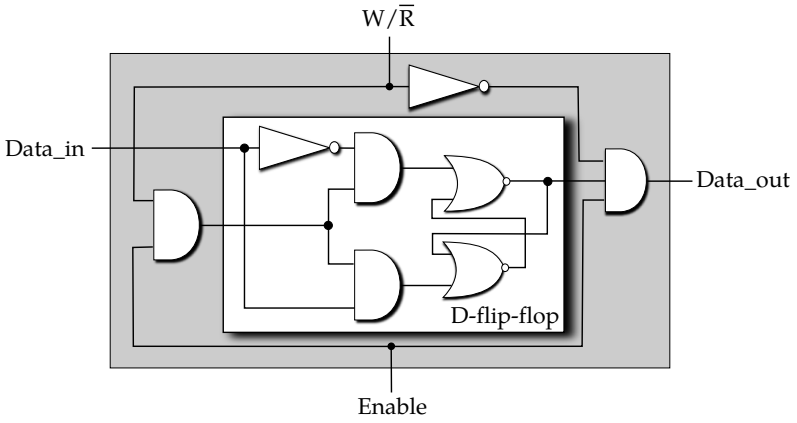

Fig. 4. Memory Cell

$$\begin{aligned} H_S(\text{memory\_cell}) &= 2 \cdot H_B(\text{Inverter}) \\ &+ 3 \cdot H_B(\text{AND}) \\ &+ 2 \cdot H_B(\text{NOR}) \\ &+ H_B(\text{AND\_2}) \\ &= 23 \cdot \log(2) \end{aligned} \tag{5}$$

The memory cells are used to build up 1-bit word arrays which compose m-bit word arrays. These word arrays are the storage matrix of the memory. The address decoders consists of a n-bit wide inverse multiplexer which itself consists of single 1-to-2 inverse multiplexers. The whole structural design hierarchy can be found in **figure 5**. The complexity for the whole memory is calculated in equation (6)

$$\begin{aligned} H_S(\text{memory}) &= H_B(address\_decoder) \\ &+ H_S(address\_decoder) \\ &+ H_B(storage\_matrix) \\ &+ H_S(storage\_matrix) \\ &= ((m+4)2^n + m^2 + 2n + m + 33) \cdot \log(2) \end{aligned} \tag{6}$$

The next part will calculate the complexity for the processor. Because of its more heterogeneous design there are more components to consider.

4.2 Processor

The design for the processor is illustrated in **figure 6**. The design of the processor is based on a MIPS design from [23]. We use a fixed address width of 32-bit and a

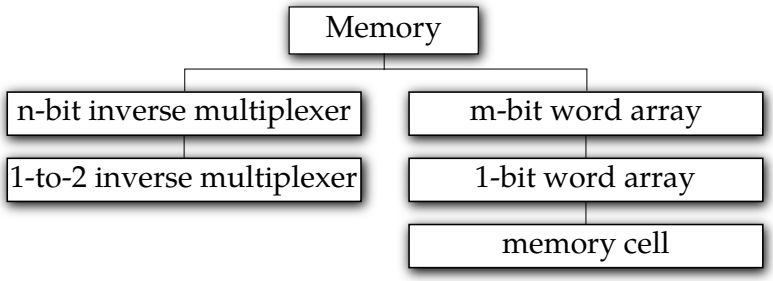

Fig. 5. Structural design hierarchy of a memory

variable data width. The processor has a separate instruction and data memory. The program counter is a m-bit wide register which can be loaded with the next address from the adder or directly with an address from an (un-)conditional jump from the control unit. The ALU can perform five basic operations (add, sub, and, or, slt).

In order to calculate the design entropy of our MIPS, the behavior entropy and structure entropy of every single component has to be calculated. As figure 6 shows, the design is very heterogeneous. Therefore, the structural design hierarchy of this processor becomes large, as shown in **figure 7**. The ALU is composed by single 1-bit ALUs. Since we already built a memory, we used this memory for the data and the instruction memory of the processor. Due to the huge amount of components and sub components, we only provide the complete complexity for the whole processor in equation (7).

$$H_S(\text{processor}) = \\ ((m+40)2^m + 2m^2 + 100m + 1748) \cdot \log(2) \tag{7}$$

4.3 The design gap

With equation (6) for the memory's entropy and (7) for the processor's entropy, both designs can be plotted as a function of the data width m. The address width for the memory is also 32-bit. As the minimum data width for the processor is 32-bit, we chart values starting with 32-bit. This leads to the complexity figures in **table 1**. The figures can then be plotted in **figure 8**.

The divergence between the processor and the memory can clearly be identified.

5 Conclusion

As figure 8 shows, the design of a processor is more complex than the design of a memory. A memory was chosen to represent highly regular structures while a

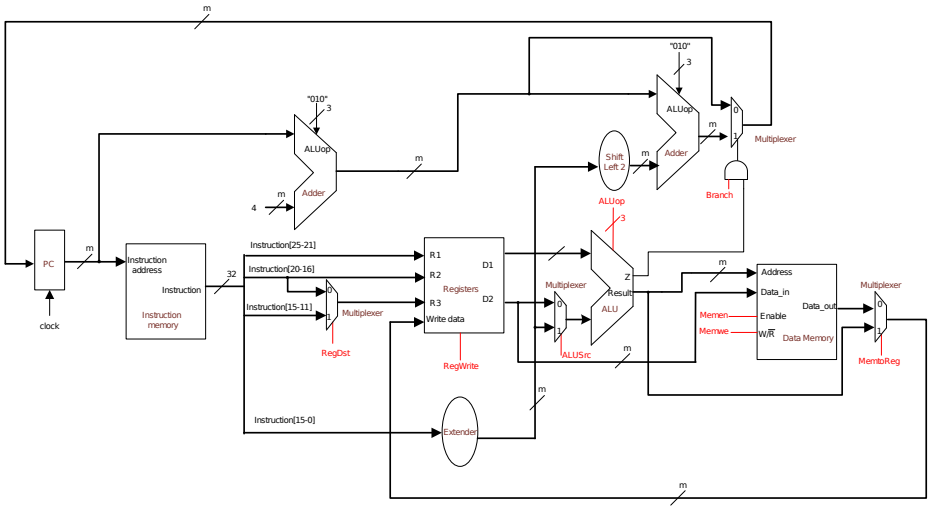

Fig. 6. Basic processor architecture

processor represents highly irregular structures. Therefore the results from the complexity calculation were expected to show that highly irregular structures are more complex than highly regular structures. In contrast to the chart in figure 2, the values for figure 8 were calculated by the complexity measurement method, which was introduced earlier.

Figures for Moore's Law and for the productivity rely on empirical data. Explanations for the slower growing productivity are missing tools, abstraction layers and design possibilities. But as this work shows, the gap is caused by the structure itself. It also shows that complexity can be calculated by relying only on key figures of a design.

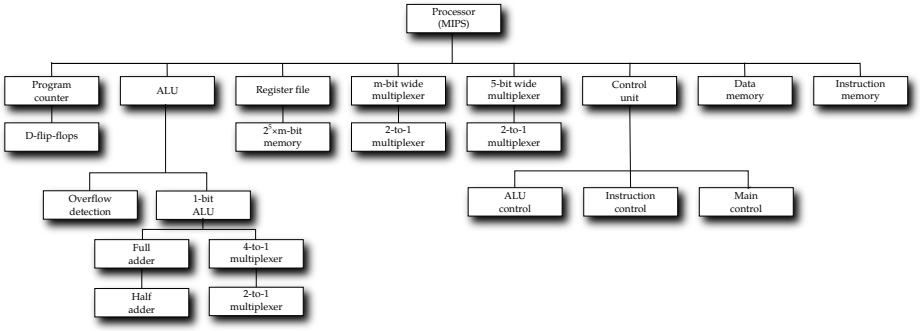

Fig. 7. Structural design hierarchy of a processor

Width	32	64	128	256
Processor	$4,3 \cdot 10^{11}$	$2,7 \cdot 10^{21}$	$8,1 \cdot 10^{40}$	$8,4 \cdot 10^{97}$
Memory	$1,5 \cdot 10^{11}$	$2,9 \cdot 10^{11}$	$5,7 \cdot 10^{11}$	$1,1 \cdot 10^{12}$

Table 1. Entropy for different data widths

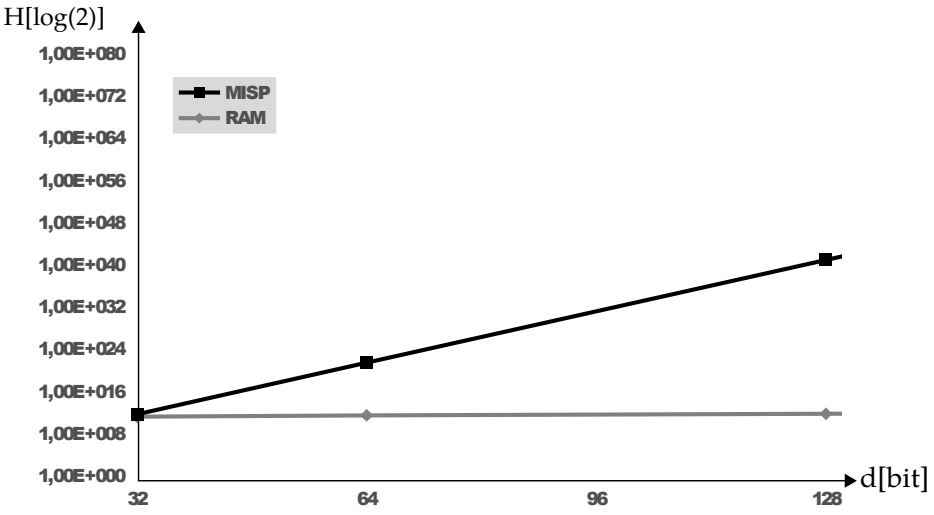

Fig. 8. Entropy of processor and memory

It wasn't necessary to take empirical data into account for the calculation. For the complexity calculation a measurement was chosen which depends on states. With this complexity calculation method, it can also be explained why the introduction of further tools, abstraction layers and design possibilities shorten the gap. Those approaches reduce the decision possibilities for designers and engineers and therefore reduces the amount of possible states.

With this work, it was demonstrated that complexity figures can be calculated. This calculation reflects empirical data known for 30 years in semiconductor industry. Therefore it reflects the effort of complexity in future designs. Concluding, that the different complexities depend on the designs' structure. Therefore the cause for the design gap also depends on the structure.

References

1. Moore, G.E.: Cramming more components onto integrated circuits. Electronics **38**(8) (April 1965)

2. Intel Corporation: Excerpts from a conversation with gordon moore: Moore's law. ftp://download.intel.com/museum/Moores_Law/Video-transcripts/Excepts_A_Conversation_with_Gordon_Moore.pdf (2005)
3. Kanellos, M.: Moore's law to roll on for another decade. http://news.cnet.com/2100-1001-984051.html (2003)
4. the INQUIRER: Gordon moore says aloha to moore's law. www.theinquirer.net/inquirer/news/1014782/gordon-moore-aloha-moore-law (2005)
5. Conference, I.S.S.C.: Isscc 2011 trends report. www.isscc.org/doc/2011/2011_Trends.pdf (2011)
6. Mollick, E.: Establishing moore's law. IEEE Annals of the History of Computing **28**(3) (2006) 62–75
7. Semiconductor Industry Association: International technology roadmap for semiconductors. www.itrs.net/Links/2007ITRS/Home2007.htm (2007)
8. Semiconductor Industry Association: International technology roadmap for semiconductors. public.itrs.net/files/1999_SIA_Roadmap/Home.htm (1999)
9. Sedcole, N.P.: Reconfigurable Platform-Based Design in FPGAs for Video Image Processing. PhD thesis (2006)
10. Henkel, J.: Closing the soc design gap. Computer **36**(9) (2003) 119–121
11. Hamilton, S.: Semiconductor research corporation: Taking moore's law into the next century. Computer **32**(1) (1999) 43–48
12. Henkel, J., Wolf, W., Chakradhar, S.: On-chip networks: A scalable, communication-centric embedded system design paradigm. VLSI Design, International Conference on **0** (2004) 845
13. Ecker, W., Müller, W., Dömer, R.: Hardware-dependent Software - Principles and Practice. Springer (2006)
14. DeMarco, T.: Controlling Software Projects: Management, Measurement, and Estimates. Prentice Hall PTR, Upper Saddle River, NJ, USA (1986)
15. Fenton, N.E., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, Revised. Course Technology (February 1998)
16. Calvano, C.N., John, P.: Systems engineering in an age of complexity: Regular paper. Syst. Eng. **7** (March 2004) 25–34
17. Hinrichs, N., Leppelt, P., Barke, E.: Building up a performance measurement system to determine productivity metrics of semiconductor design projects. In IEEE, ed.: IEEE International Engineering Management Conference (IEMC), Austin TexasErmolayev2007, IEEE (2007) CD–ROM Proceedings
18. Menhorn, B., Slomka, F.: Design entropy concept. In: ESWEEK 2011 Compilation Proceedings. (10 2011)
19. Menhorn, B., Slomka, F.: States and complexity. In Dumitrescu, D., Lung, R.I., Cremene, L., eds.: Coping with Complexity COPCOM 2011. (10 2011) 68–88
20. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27** (1948) 379–423, 623–656
21. Tolman, R.C.: The principles of statistical mechanics. Oxford Univ. Pr., London (1938)
22. Menhorn, B., Slomka, F.: Project Management Through States. In: IEMS 2009: International Conference on Engineering Management and Service Sciences. (2009)
23. Patterson, D.A., Hennessy, J.L.: Computer Organization and Design: The Hardware/software Interface. 3 edn. Morgan Kaufmann (2005)