Reversible Circuits: Recent Accomplishments and Future Challenges for an Emerging Technology*

(Invited Paper)

Rolf Drechsler[1,2] and Robert Wille[1]

[1] Institute of Computer Science, University of Bremen
Group of Computer Architecture, D-28359 Bremen, Germany
{drechsle,rwille}@informatik.uni-bremen.de
[2] Cyber-Physical Systems
DFKI GmbH, D-28359 Bremen, Germany
rolf.drechsler@dfki.de

Abstract. Reversible circuits build the basis for emerging technologies like quantum computation and have promising applications in domains like low power design. Hence, much progress in the development of design solutions for this kind of circuits has been made in the last decade. In this paper, we provide an overview on reversible circuits as well as their applications. We discuss recent accomplishments and, finally, have a look on future challenges in the design of circuits for this emerging technology.

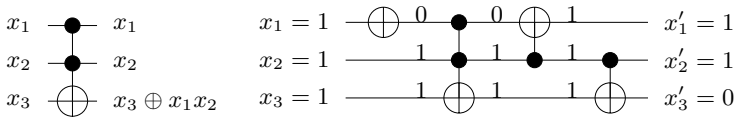
1 Introduction

Reversible circuits represent an emerging technology based on a computation paradigm which significantly differs from conventional circuits. In fact, they allow bijective operations only, i.e. n-input n-output functions that map each possible input vector to a unique output vector. Reversible computation enables several promising applications and, indeed, superiors conventional computation paradigms in many domains including but not limited to quantum computation or low power design (see e.g. [1–3]). But since reversible logic is subject to certain characteristics and restrictions, the design methodology of circuits and systems following the reversible computation paradigm significantly differs from the established (conventional) design flow.

However, much progress in the development of design solutions for reversible circuits has been made in the last decade: In particular, methods for synthesis have been developed (see e.g. [4]). These got manifested in first publicly available design tools [5] and enabled the design of initial circuit representations available at benchmark libraries [6]. Encouraged by these fundamental work, researchers now are striving for more scalability – a first hardware description language already is available [7]. These promising results build the basis for the development of a design flow which is competitive to the design of conventional circuits.

In this paper, we provide an overview on reversible circuits as well as their applications. We discuss recent accomplishments and, finally, have a look on future challenges in the design of circuits for this emerging technology. References to the respective original work are provided for further reading.

* This work was supported by the German Research Foundation (DFG; DR 287/20-1).



(a) Toffoli gate (b) Toffoli circuit

Fig. 1. Toffoli gate and Toffoli circuit

2 Reversible Circuits

A Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$ over the variables $X := \{x_1, \ldots, x_n\}$ is *reversible* iff (1) its number of inputs is equal to the number of outputs (i.e. $n = m$) and (2) it maps each input pattern to a unique output pattern. Reversible functions are realized by reversible circuits. A *reversible circuit* G is a cascade of reversible gates, where fanout and feedback are not directly allowed [1]. Each variable of the function f is thereby represented by a *circuit line*, i.e. a signal through the whole cascade structure on which the respective computation is performed. Computations are performed by *reversible gates*. In the literature, reversible circuits composed of *Toffoli gates* are frequently used. A Toffoli gate is composed of a (possibly empty) set of *control lines* $C = \{x_{i_1}, \ldots, x_{i_k}\} \subset X$ and a single *target line* $x_j \in X \setminus C$. The Toffoli gate inverts the value on the target line if all values on the control lines are assigned to 1 or if $C = \emptyset$, respectively. All remaining values are passed through unaltered.

Fig. 1(a) shows a Toffoli gate drawn in standard notation, i.e. control lines are denoted by ●, while the target line is denoted by $\oplus$. A circuit composed of several Toffoli gates is depicted in Fig. 1(b). This circuit maps e.g. the input 111 to the output 110 and vice versa.

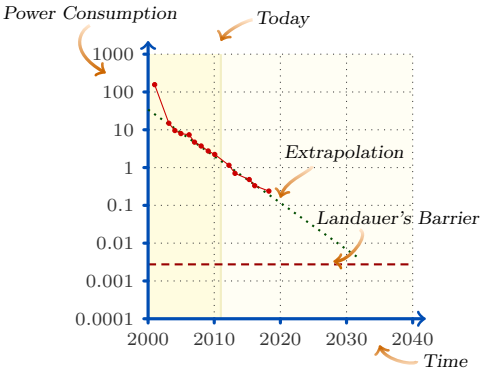
3 Applications of Reversible Circuits

Several promising applications of reversible circuits have been shown – with quantum computation and low power design being the most prominent examples. In this section, we briefly review possible research areas which are advanced by reversible circuits.

3.1 Quantum Computation

In a quantum computer [1], information is represented in terms of *qubits* instead of bits. In contrast to Boolean logic, qubits do not only allow to represent Boolean 0's and Boolean 1's, but also the superposition of both. In other words, using quantum computation and qubits in superposition, functions can be evaluated with different possible input assignments in parallel. Unfortunately, it is not possible to obtain the current state of a qubit. Instead, if a qubit is measured, either 0 or 1 is returned depending on a respective probability.

Nevertheless, using these quantum mechanical phenomena, quantum computation allows for breaching complexity bounds which are valid for computing devices based on conventional mechanics. The Grover search [8] and the factorization algorithm by Shor [9] rank among the most famous examples for quantum algorithms that solve problems in time complexities which cannot be achieved using conventional computing. The first algorithm addresses thereby the search of an item in an unsorted database with k items in time $O(\sqrt{k})$, whereas conventional methods cannot be performed using less than linear time. Shor's algorithm performs prime factorization in polynomial time, i.e. the algorithm is exponentially faster than its best known conventional counterpart. First physical realizations of quantum circuits have been presented e.g. in [10].



The figure illustrates the development of the power consumption of an elementary computational step in recent CMOS generations (based on values from [14]). The power consumption is thereby determined by CV_t^2, where V_t is the threshold voltage of the transistors and C is the total capacitance of the capacitors in the logic gate. The capacitance C is directly proportional to $\frac{LW}{t}$, i.e. to the length L and the width W of the transistors. Reducing these sizes of transistors enables significant reductions in the power consumption as shown in the extrapolation. However, this development will reach a fundamental limit when power consumption is reduced to $k \cdot T \cdot \log(2)$ Joule.

Fig. 2. Power consumption Q in different CMOS generations

Reversible circuits are of interest in this domain since all quantum operations inherently are reversible. Since most of the known quantum algorithms include a large Boolean component (e.g. the database in Grover's search algorithm and the modulo exponentiation in Shor's algorithm), the design of these components is often conducted by a two-stage approach, i.e. (1) realizing the desired functionality as a reversible circuit and (2) map the resulting circuit to a functionally equivalent quantum circuit (using methods introduced e.g. in [11, 12]).

3.2 Low Power Design

Pioneering work by Landauer [13] showed that, regardless of the underlying technology, loosing information during computation causes power dissipation. More precisely, for each "lost" bit of information, at least $k \cdot T \cdot \log(2)$ Joules are dissipated (where k is the Boltzmann constant and T is the temperature). Since today's computing devices are usually built of elementary *gates* like `AND`, `OR`, `NAND`, etc., they are subject to this principle and, hence, dissipate this amount of power in each computational step.

Although the theoretical lower bound on power dissipation still does not constitute a significant fraction of the power consumption of current devices, it nonetheless poses an obstacle for the future. Fig. 2 illustrates the development of the power consumption of an elementary computational step in recent and expected future CMOS generations (based on values from [14]). The figure shows that today's technology is still a factor of 1,000 away from the Landauer limit and that the expected CMOS development will reduce this to a factor of 100 within the next 10 years. However, a simple extrapolation also shows that the trend cannot continue with the current family of static CMOS gates as no amount of technological refinement can overcome the Landauer barrier. Moreover, the Landauer limit is only a *lower* bound on the dissipation. Gershenfeld has shown that the actual power dissipation corresponds to the amount of power used to represent a signal [15], i.e. Landauer's barrier is closer than immediately implied by the extrapolation from Fig. 2.

Since reversible circuits bijectively transforms data at each computation step, the above-mentioned information loss and its resulting power dissipation does not occur. Because of this, reversible circuits manifests themselves as the only way to break this fundamental limit. In fact, it has been proven that to enable computations with no power consumption at all, the underlying realization must follow reversible computation principles [16]. These fundamental results motivate researchers in investigating this direction further. First physical realizations have been presented e.g. in [2].

3.3 Further Applications

While quantum computation and low power design represent the most prominent application areas of reversible circuits, promising results have been achieved in other domains as well.

An obvious filed is for example the design of encoding and decoding devices. Since encoders and decoders always realize reversible one-to-one mappings, the application of reversible circuits is a reasonable choice. However, so far, most of such devices are implemented in a conventional, i.e. irreversible, manner and, therefore, miss potential benefits in their design. An exception provides the approach recently presented in [3] where, for the first time, encoders and decoders for the on-chip communication between different components of a system-on-a-chip have been designed by means of reversible circuits.

As another domain, adiabatic circuits [17] utilize signals that, in order to avoid power losses, switch their states very slowly. When the power dissipation from switching transitions has been suppressed to a minimum, the static power dissipation caused by leaking devices in advanced, extremely miniaturized process technologies will become very substantial. Regardless of the computing paradigm, the static energy is present in virtually all transistor circuits. However, reversible circuits have the advantage that they naturally are suited for adiabatic switching without any extra circuitry.

Finally, program inversion provides a proper application for reversible computation. Motivated e.g. through debugging purposes, the question how to automatically derive the inverse of a given program is addressed. As most of the existing programs follow the conventional, i.e. irreversible, computation paradigm, program analysis techniques (e.g. [18]) or interpretive solutions (e.g. [19]) are applied so far. However, programs based on a reversible, i.e. invertible, computation paradigm would allow an inherent and obvious program inversion.

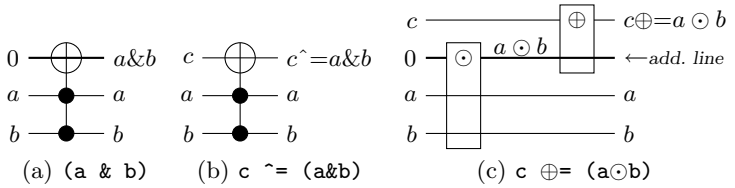
4 Design of Reversible Circuits

Motivated by the promising applications outlined above, design and synthesis of reversible circuits became an active research area where a significant amount of approaches has been presented over the last years. This section provides an overview of the initial contributions and the recent accomplishments in this area. Based on this, future challenges and possible solutions to them are discussed.

4.1 The Beginning

First approaches for synthesis of reversible circuits relied on simple function representations such as permutations or truth tables. Somewhat later, more compact data-structures like decision diagrams, positive-polarity Reed-Muller expansion, or Reed-Muller spectra have been utilized for this purpose. Complementary, also exact synthesis approaches, i.e. methods ensuring minimality of the resulting circuits, have been proposed. An overview of these methods is e.g. provided in [4]. The given functions need thereby to be reversible. Since this is not the case for many practically relevant functions, a pre-processing step often is performed first which embeds the desired non-reversible functionality into a proper reversible function [20, 21].

Until today, these approaches build the basis of the design of reversible circuits and can be used e.g. to realize basic building blocks for reversible computations. However, the scalability of all these approaches is limited. That is, the methods are applicable for relatively small functions, i.e. functions with at most 30 variables only. Hence, after the initial development of synthesis methods for this upcoming emerging technology, researchers began to strive for more scalable design solutions.



(a) `(a & b)` (b) `c ^= (a&b)` (c) `c ⊕= (a⊙b)`

Fig. 3. Circuits obtained by HDL-based synthesis

4.2 Recent Accomplishments

While being an open research problem for many years, how to automatically realize large functions and more complex logic as reversible circuit has been tackled by recently presented solutions. More compact function representations like *Binary Decision Diagrams* (BDDs) [22] have been applied for this purpose. In contrast to the synthesis approaches outlined above, a hierarchical synthesis scheme is thereby applied. That is, the (possibly very large) function to be synthesized, is decomposed into smaller sub-functions. The decomposition is repeatedly applied until a sub-function results for which a building block exists. By accordingly composing these building blocks, a circuit representing the desired function results.

The development of this paradigm in the domain of reversible circuit design eventually led to the definition and consideration of a *Hardware Description Language* (HDL) for reversible circuits in [7]. In order to ensure reversibility in the description, this HDL distinguishes thereby between reversible operations (denoted by $\oplus=$) and not necessarily reversible *binary operations* (denoted by $\odot$). The former class of operations assigns values to a signal on the left-hand side. Therefore, the left-hand side signal must not appear in the expression on the right-hand side. Furthermore, only a restricted set of assignment operations exists, namely increase (`+=`), decrease (`-=`), and bit-wise XOR (`^=`). These operations preserve the reversibility (i.e. it is possible to compute these operations in both directions). In contrast, binary operations, e.g. arithmetic, bit-wise, logical, or relational operations, may not be reversible. Thus, they can only be used in right-hand expressions which preserve the values of the respective inputs. In doing so, all computations remain reversible since the input values can be applied to reverse any operation. For example, to describe a multiplication (i.e. `a*b`), a new free signal `c` must be introduced which is used to store the product (i.e. `c^=a*b` is applied). In comparison to common (non-reversible) languages, this forbids statements like `a=a*b`.

Having an HDL description like this, automatic synthesis of complex reversible circuits became applicable for the first time. Again, a hierarchical synthesis paradigm is thereby applied [7]. That is, existing realizations of the individual operations are combined so that the desired circuit is realized.

While this represents a significant step towards design of complex reversible circuits, the resulting realizations suffer from having a very large number of additional circuit signals. This is caused by the fact that building blocks representing non-reversible operations usually require additional circuit lines with constant inputs 0. For example, the `AND` operation (`a & b`) – a typical binary operation – is non-reversible. In order to realize the `AND` nevertheless, an additional circuit line with a constant input is required as shown in Fig. 3(a). However, since binary operations can only be applied in combination with a reversible assignment operation, these lines principally are not necessary (e.g. the overall statement `c ^= (a & b)` can be realized without additional lines as shown in Fig. 3(b)). But determining the respective circuits for arbitrary combinations of reversible assignment operations and binary operations is a cumbersome task.

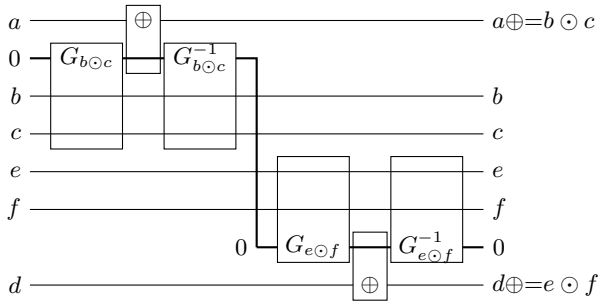


Fig. 4. Improved HDL-based synthesis

Hence, so far HDL synthesizers realize reversible circuits using the hierarchical scheme as illustrated in Fig. 3(c) for a general statement `c ⊕= (a ⊙ b)`[3]. That is, first the respective building block(s) for binary operations are solely applied. This requires additional circuit lines. Afterwards, the intermediate results from the binary operations (buffered in the additional circuit lines) are applied together with the building block of the corresponding reversible assignment operation.

4.3 Future Challenges

With the recent accomplishments, the scalability of synthesis approaches for reversible circuits has significantly been improved. Although further improvements and extensions concerning the available descriptions means are still desired (e.g. more complex data-types, support of sequential behavior, etc.), the specification and realization of complex reversible circuits at a high level became possible. But as outlined above, the resulting circuits still suffer from having too many circuit lines. A first approach addressing this issue is already available [23]. However, as e.g. shown in [24], these promising results are still far away from the optimum. Hence, after progress concerning the scalability has been made, how to reduce the number of additional circuit lines remains an open problem.

A possible direction to solve this issue might be to "undo" the values of intermediate results once they are not needed any longer. Then, no new additional lines might be required to buffer upcoming intermediate results. Instead, the existing (reset) signals can be re-used. The general idea is briefly illustrated in Fig. 4 by means of the following two generic HDL statements:

```
a ⊕= (b ⊙ c);
d ⊕= (e ⊙ f);
```

First, two sub-circuits $G_{b\odot c}$ and $G_{a\oplus=b\odot c}$ are added ensuring that the first statement is realized. This is equal to the established procedure from Fig. 3(c) and leads to additional lines with constant inputs. But in contrast to existing HDL synthesizers, a further sub-circuit $G^{-1}_{b\odot c}$ is applied afterwards. Since $G^{-1}_{b\odot c}$ is the inverse of $G_{b\odot c}$, this sets the circuit lines buffering the result of $b\odot c$ back to the constant 0. As a result, these circuit lines can be reused in order to realize the following statements as illustrated for $d\oplus{=}e\odot f$ in Fig. 4.

Following this procedure, significant improvements can be achieved. However, even this solution does not entirely reduce the total number of additional lines to none. Hence, a further consideration is necessary. While this represents one of the major obstacles for synthesis of complex reversible circuits today, beyond that also the following challenges should be addressed:

[3] Circuit lines drawn through the blocks represent thereby signals whose values are preserved.

– Quantum cost [11, 12] are mainly applied as cost metric to evaluate the synthesized reversible circuits so far. But beyond that, also other, more technology-specific constraints should be considered (e.g. transistor cost [25] or nearest-neighbor requirements [26, 27]).
– In order to realize practical reversible circuits, sequential behavior has to be supported. Initial work considering this issue can be found e.g. in [28–30] but requires more research to become applicable for complex designs.
– Followed by the increasing power of the synthesis methods, also new verification issues will emerge. Hence, developing appropriate checkers particularly for complex reversible designs is a logical next step. Researchers can thereby build on first results achieved for equivalence checking (see e.g. [31, 32]) and even debugging [33].
– Furthermore, questions related to test of reversible circuits more and more becomes of interest. A basis builds the work presented e.g. in [34, 35]. However, with ongoing progress in the development of further (and larger) physical realizations, new test models and ATPG approaches are needed.
– Finally, all these methods and approaches have to be combined to an integrated design flow. Although first simple flows are provided e.g. through tools like RevKit [5], this remains the long-term challenge of research in the domain of reversible circuit design.

5 Conclusions

Reversible circuits is an emerging technology with promising applications. In the last decade, synthesis of reversible circuits has intensely been studied and impressive accomplishments have been made. Starting from first synthesis approaches applicable to functions represented by permutations or truth tables, today it is possible to design and synthesize complex circuits using a reversible hardware description language. However, HDL-based synthesis still suffers from the fact that circuits with a large number of lines are generated. This represents a major challenge for the future. In this paper, a brief overview on reversible circuits and their applications, recent accomplishment, as well as future challenges has been provided.

References

1. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge Univ. Press (2000)
2. Berut, A., Arakelyan, A., Petrosyan, A., Ciliberto, S., Dillenschneider, R., Lutz, E.: Experimental verification of landauer's principle linking information and thermodynamics. Nature **483** (2012) 187–189
3. Wille, R., Drechsler, R., Oswald, C., Garcia-Ortiz, A.: Automatic design of low-power encoders using reversible circuit synthesis. In: Design, Automation and Test in Europe. (2012) 1036–1041
4. Drechsler, R., Wille, R.: From truth tables to programming languages: Progress in the design of reversible circuits. In: Int'l Symp. on Multi-Valued Logic. (2011) 78–85
5. Soeken, M., Frehse, S., Wille, R., Drechsler, R.: RevKit: An Open Source Toolkit for the Design of Reversible Circuits. In: Reversible Computation 2011. Volume 7165 of Lecture Notes in Computer Science. (2012) 64–76 RevKit is available at www.revkit.org.
6. Wille, R., Große, D., Teuber, L., Dueck, G.W., Drechsler, R.: RevLib: an online resource for reversible functions and reversible circuits. In: Int'l Symp. on Multi-Valued Logic. (2008) 220–225 RevLib is available at http://www.revlib.org.
7. Wille, R., Offermann, S., Drechsler, R.: SyReC: A programming language for synthesis of reversible circuits. In: Forum on Specification and Design Languages. (2010) 184–189

8. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Theory of computing. (1996) 212–219
9. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. Foundations of Computer Science (1994) 124–134
10. Vandersypen, L.M.K., Steffen, M., Breyta, G., Yannoni, C.S., Sherwood, M.H., Chuang, I.L.: Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. Nature **414** (2001) 883
11. Barenco, A., Bennett, C.H., Cleve, R., DiVinchenzo, D., Margolus, N., Shor, P., Sleator, T., Smolin, J., Weinfurter, H.: Elementary gates for quantum computation. The American Physical Society **52** (1995) 3457–3467
12. Miller, D.M., Wille, R., Sasanian, Z.: Elementary quantum gate realizations for multiple-control toffolli gates. In: Int'l Symp. on Multi-Valued Logic. (2011) 288–293
13. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5** (1961) 183
14. Zeitzoff, P., Chung, J.: A perspective from the 2003 ITRS. IEEE Circuits & Systems Magazine **21** (2005) 4–15
15. Gershenfeld, N.: Signal entropy and the thermodynamics of computation. IBM Systems Journal **35**(3-4) (1996) 577–586
16. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev **17**(6) (1973) 525–532
17. Patra, P., Fussell, D.: On efficient adiabatic design of MOS circuits. In: Workshop on Physics and Computation, Boston (1996) 260–269
18. Glück, R., Kawabe, M.: A method for automatic program inversion based on LR(0) parsing. Fundamenta Informaticae **66**(4) (01 2005) 367–395
19. Abramov, S., Glück, R.: The universal resolving algorithm and its correctness: inverse computation in a functional language. Science of Computer Programming **43**(23) (2002) 193 – 229
20. Maslov, D., Dueck, G.W.: Reversible cascades with minimal garbage. IEEE Trans. on CAD **23**(11) (2004) 1497–1509
21. Miller, D.M., Wille, R., Dueck, G.: Synthesizing reversible circuits for irreversible functions. In: EUROMICRO Symp. on Digital System Design. (2009) 749–756
22. Wille, R., Drechsler, R.: BDD-based synthesis of reversible logic for large functions. In: Design Automation Conf. (2009) 270–275
23. Wille, R., Soeken, M., Drechsler, R.: Reducing the number of lines in reversible circuits. In: Design Automation Conf. (2010) 647–652
24. Wille, R., Keszöcze, O., Drechsler, R.: Determining the minimal number of lines for large reversible circuits. In: Design, Automation and Test in Europe. (2011)
25. Thomson, M.K., Glück, R.: Optimized reversible binary-coded decimal adders. J. of Systems Architecture **54** (2008) 697–706
26. Khan, M.H.A.: Cost reduction in nearest neighbour based synthesis of quantum boolean circuits. Engineering Letters **16** (2008) 1–5
27. Saeedi, M., Wille, R., Drechsler, R.: Synthesis of quantum circuits for linear nearest neighbor architectures. Quantum Information Processing **10**(3) (2011) 355–377
28. Chuang, M., Wang, C.: Synthesis of reversible sequential elements. In: ASP Design Automation Conf. (2007) 420 –425
29. Nayeem, N.M., Hossain, M.A., Jamal, L., , Babu, H.: Efficient design of shift registers using reversible logic. In: Int'l Conf. on Signal Processing Systems. (2009) 474–478
30. Himanshu, H., Ranganathan, N.: Design of reversible sequential circuits optimizing quantum cost, delay, and garbage outputs. J. Emerg. Technol. Comput. Syst. **6** (2010) 14:1–14:31
31. Viamontes, G.F., Markov, I.L., Hayes, J.P.: Checking equivalence of quantum circuits and states. In: Int'l Conf. on CAD. (2007) 69–74
32. Wille, R., Große, D., Miller, D.M., Drechsler, R.: Equivalence checking of reversible circuits. In: Int'l Symp. on Multi-Valued Logic. (2009) 324–330
33. Wille, R., Große, D., Frehse, S., Dueck, G.W., Drechsler, R.: Debugging of Toffoli networks. In: Design, Automation and Test in Europe. (2009) 1284–1289
34. Polian, I., Fiehn, T., Becker, B., Hayes, J.P.: A family of logical fault models for reversible circuits. In: Asian Test Symp. (2005) 422–427
35. Wille, R., Zhang, H., Drechsler, R.: ATPG for reversible circuits using simulation, Boolean satisfiability, and pseudo Boolean optimization. In: IEEE Annual Symposium on VLSI. (2011) 120–125