COMPUTING THE VAN DER WAERDEN NUMBER $W(3,4) = 293$

Michal Kouril, PhD
Cincinnati Children's Hospital Medical Center,
University of Cincinnati, Cincinnati, OH
Michal.Kouril@cchmc.org

Received: 3/26/12, Revised: 6/30/12, Accepted: 8/24/12, Published: 9/3/12

Abstract

We have verified that the van der Waerden number $W(3,4)$ is 293, that is, 293 is the smallest integer $n = W(3,4)$ such that whenever the set of integers $\{1,2,\ldots,n\}$ is 3-colored, there exists a monochromatic arithmetic progression (a.p.) of length 4. The fact that $W(3,4) > 292$ was established by Rabung in 1979, who constructed a 3-coloring of $\{1,2,\ldots,292\}$ without a monochromatic a.p. of length 4 using Folkman's method. By a combination of novel preprocessing and efficient solver design we have determined all 3-colorings of $\{1,2,\ldots,292\}$ that do not contain a monochromatic a.p. of length 4, and have shown that none of them can be extended to $\{1,2,\ldots,293\}$ without generating a monochromatic a.p. of length 4 (thus $W(3,4) = 293$). The search was similar to what we previously used to establish the result $W(2,6) = 1132$. The search space was split into 3,642,579 non-equivalent initial colorings of length 19 during preprocessing, and these colorings were then processed using an FPGA-based solver. Subsequent post-processing recovered the entire search space and produced our result. We also calculated a number of new off-diagonal van der Waerden numbers, and verified all currently known numbers.

1. Introduction

In 1926 B.L. Van der Waerden proved the following result[19]: given positive integers K and L, both having value at least two (one being trivial), there is a smallest integer $n = W(K,L)$ such that every K-coloring of $\{1,2,\ldots,n\}$ contains a monochromatic a.p. of length L. The original proof gave bounds on $W(K,L)$ that were huge and not even primitive recursive. Shelah, in 1988 [17], obtained primitive recursive bounds that were much lower but still quite large. Gowers, in 2001, obtained even better bounds [11]; however, they are still impractical for computing exact numbers. The empirical evidence (such as it is) seems to indicate that

$K\backslash L$	3	4	5	6
2	9	35	178	1132
3	27	293		
4	76			

Table 1: van der Waerden numbers

the numbers $W(K, L)$ are in reality far smaller than the three proofs cited would indicate. The results in this paper further that evidence.

The currently known Van der Waerden numbers $W(K, L)$ are shown in Table 1. The bound $W(3, 4) > 292$ was previously shown by Rabung in 1979 using Folkman's method [15]. We show in this paper that this bound, surprisingly, is actually sharp, i.e., $W(3, 4) = 293$.

2. Overview of the $W(3, 4) = 293$ Computational Proof

Our proof that $W(3, 4) = 293$ follows closely the computation of $W(2, 6) = 1132$ given in [13]. We begin with a preprocessing step starting with an unavoidable pattern $\{112\}$. More precisely, unavoidable means that for a sufficiently large n, if $\{1, \ldots, n\}$ is 3-colored with colors $\{1, 2, 4\}$ and doesn't contain a monochromatic a.p. of length 4, then there is an integer x such that the coloring of $x, x+1, x+2$ is $1, 1, 2$ or something similar (e.g., $2, 2, 1$ or $4, 4, 2$, etc.). (The reason that the colors are encoded in a power of two sequence is related to the internal representation of the integers in our solvers, which works by narrowing down the list of possible colorings of a given integer.)

We expand the unavoidable pattern $\{112\}$ by eliminating redundant patterns thereby significantly cutting down the search space. This expansion continues while the pattern set is reasonably large, stopping at 3,642,579 initial patterns.

These initial patterns are then processed one by one on a set of Field Programmable Gate Arrays (FPGAs) housing a special SAT solver. Each pattern is padded with unassigned variables on each side, effectively placing the initial pattern in the middle and the SAT solver attempts to assign colors to the unassigned variables avoiding any monochromatic a.p. length 4. Using standard SAT terminology we call such an assignment a *satifying assignment*. Such an assignment exists for only a fraction of the initial patterns. The FPGA solver does not return the assignment itself, but rather this is done on a computational cluster with another SAT solver able to report all satisfying assignments.

These satisfying assignments are then post-processed to recover the original search space and extended to find a set of longest 3-colored patterns without containing a monochromatic a.p. of length 4. We found that the maximum length of such patterns is 292, thereby proving that $W(3,4) = 293$.

3. Preprocessing

3.1. The Unavoidable Pattern {112}

As indicated earlier, we have started the preprocessing step with the unavoidable pattern {112}. This pattern was obtained by adding additional constraints to the $W(3,4)$ solver, which not only eliminated any colorings containing a monochromatic a.p. of length 4, but also any colorings that contained two consecutive numbers with the same color ({11}, {22}, or {44}). By adding additional constraints into our solver we found out that the minimum n for which 112 is unavoidable is 94.

3.2. Expansion Operations

As discussed in [13], we used two elementary operations to find equivalent patterns and cut down the search space. As mentioned earlier, for ease of implementation we labeled the three colors as 1,2,4. The operations leading to equivalence classes of patterns were:

1. re-coloration:
   e.g., $\{112\}$ is equivalent to $\{221, 224, 441, 442, 114\}$.
2. reversal
   e.g., $\{112\}$ is equivalent to $\{211\}$.

These operations can be applied to any pattern. In the re-coloration preprocessing step we went through iterations where we kept extending a set of patterns by one additional element, and then transformed them to the unique lowest value in their equivalence class. The lowest value in an equivalence class of patterns is defined as the pattern that represents the smallest base ten number in the class. For example, 112 is the lowest value since $\{221, 224, 441, 442, 114, 122, 144, 211, 244, 411, 422\}$ all have a base ten number value higher than 112.

Our starting point for the preprocessing step was the unavoidable pattern {112}, which led to 3,642,579 satisfying assignments of length 19 following the expansion steps.

The first few expansions are:

- Length = 4, number of patterns = 3

  $$\{1121, 1122, 1124\}$$

- Length = 5, number of patterns = 9

$$\{1121\} \rightarrow \{11211, 11212, 11214\} \rightarrow \{11211, 11212, 11214\}$$
$$\{1122\} \rightarrow \{11221, 11222, 11224\} \rightarrow \{11221, 11122, 11224\}$$
$$\{1124\} \rightarrow \{11241, 11242, 11244\} \rightarrow \{11241, 11242, 11244\}$$

$$\{11122, 11211, 11212, 11214, 11221, 11224, 11241, 11242, 11244\}$$

- Length $= 6$, number of patterns $= 26$

$$\{111211, 111221, 111222, 111224, 111244, 112112, 112114, 112121, 112122, 112124, 112141, 112142, 112144, 112211, 112212, 112214, 112241, 112242, 112244, 112411, 112412, 112414, 112421, 112424, 112441, 112442\}$$

The following table (Table 2) lists the number of patterns at each step $1 \ldots 19$. Although the rate of growth is decreasing, the absolute number of patterns is quite large, and we stop expanding at what we consider is a still manageable number.

Table 2: Number of patterns at each preprocessing step $1 \ldots 19$

Length of the patterns	Number of patterns
4	3
5	9
6	26
7	68
8	178
9	459
10	1,148
11	2,880
12	7,235
13	17,805
14	43,969
15	109,608
16	262,714
17	638,988
18	1,548,112
19	3,642,579

4. FPGA-Based SAT Solver

In order to process all 3,642,579 patterns generated in the previous step, we built an FPGA-based solver, which, as shown in [13], is significantly faster than the

PC-based solver.

An FPGA solver is given an initial assignment, which contains one preprocessed pattern placed in the middle of n unassigned variables. As discussed above, the length of the preprocessed patterns was 19. For example, when $n = 160$ there are 141 unassigned variables, 70 placed before and 71 after the pattern. In the first pass all 3,642,579 patterns were processed using a solver with $n = 160$. After about a month of computation, the FPGA solver found that there were exactly 1408 initial patterns that could be extended to patterns of length 160 not containing a monochromatic a.p. of length 4. In order to reduce the number of patterns further, we increased our pattern length to 200 by adding unassigned variables on both sides of these assignments. After a few hours of additional computation, it turned out that exactly 23 initial patterns could be extended to patterns of length 200 not containing a monochromatic a.p. of length 4.

5. Postprocessing

For 23 patterns from the previous phase, we collected 420 satisfying assignments for $n = 200$ by adding unassigned variables and using the SAT solver to determine all solutions.

In order to recover the entire search space we first re-colored all patterns of length 200 with a permutation of all 3 colors and then shifted the resulting set in the space of length $n = 292$. Then we used the SAT solver to fill in the unassigned variables and obtained 468 unique patterns (called "extreme" patterns) representing all possible colorings of length $n = 292$ without a monochromatic a.p. of length 4. None of these colorings extend further, so that $W(3; 4) = 293$.

All extreme patterns have the following structure:

$$tPP_{rev}uPP_{rev}vPP_{rev}w$$

where

- $P = 122424214441244241114142441121241142224221221114$ (length 48);
- P_{rev} is a reversal of the pattern P;
- t, u, v, w are "glue" variables.

Glue variables can have $3^4 - 3 = 78$ values, the only constraint being avoiding an assignment where all glue variables have the same color. Additional re-coloration provides a total of 6 combinations of the pattern P ,hence 468 unique patterns.

6. FPGA-Based and PC-Based SAT Solver Design Notes

The FPGA solver design used in [13] was focused only on 2-colored van der Waerden numbers, which allowed us to simplify (and save the much needed FPGA area). For a description of the original version please refer to [13].

We further developed the solver and added two additional designs. Here is the summary of all three designs:

- Original version for $K = 2$ and one inference block;
- Original version expanded to K colors but maintaining only one inference block;
- K-inference blocks version.

All three versions maintained the basic schema (see Figure 1) and realized tradeoffs in speed, occupied space on an FPGA chip, ease of routing for the synthesis tool, and so forth. The original version was still best[1] for $K = 2$ numbers with uniform bounds $W(2, L)$. The second version of the solver was best for targets $K > 2$ and uniform bounds. The third version was the only version on which a computation of van der Waerden numbers $W(K; L_1, L_2, \ldots L_K)$ with nonuniform bounds in each color was possible (due to the existence of an inference block for each color). Recall that $W(K; L_1, L_2, \ldots L_K)$ is the smallest integer $n = W(K; L_1, L_2, \ldots L_K)$ such that every K-coloring of $1, 2, \ldots, n$ contains a monochromatic a.p. of length L_i for some i in $\{1, 2, \ldots, K\}$.

6.1. FPGA Solver Design: Original Version Expanded to K Colors

We expanded the design of the $W(2, 6)$ solver to accommodate multiple colors. The schema remained unchanged (see Figure 1). Just as with $W(2, 6)$ we chose to utilize only single inference block (vs. K-blocks), which is a block searching for inferences and detects contradictions in a single color at a time. The idea of using single inference blocks rather than K-blocks was related to the size of an inference block, which was by far the largest part of the combinatorial portion of the solver. This saved FPGA space and allowed for multiple solvers to be placed on a single chip (up to 4 on LX110T).

When computing $W(3, 4)$ we had 9 FPGA boards available: 4 boards with Xilinx Virtex 4 (LX60) and 5 boards with Xilinx Virtex 5 (LX110T).

The combinatorial block shown in Figure 2 shows the complexity of the solver. The entire combinatorial block was verified by comparing its formal definition in the cryptol language [1] and VHDL code using the cryptol equivalence-checking tool.

[1] We define *best* as the combination of achievable clock speed and solvers per area allowing to process the given patterns fastest.

Figure 1: FPGA solver – basic schema

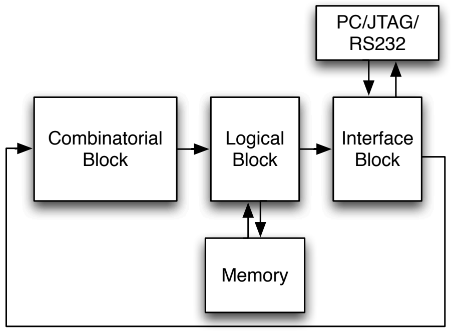
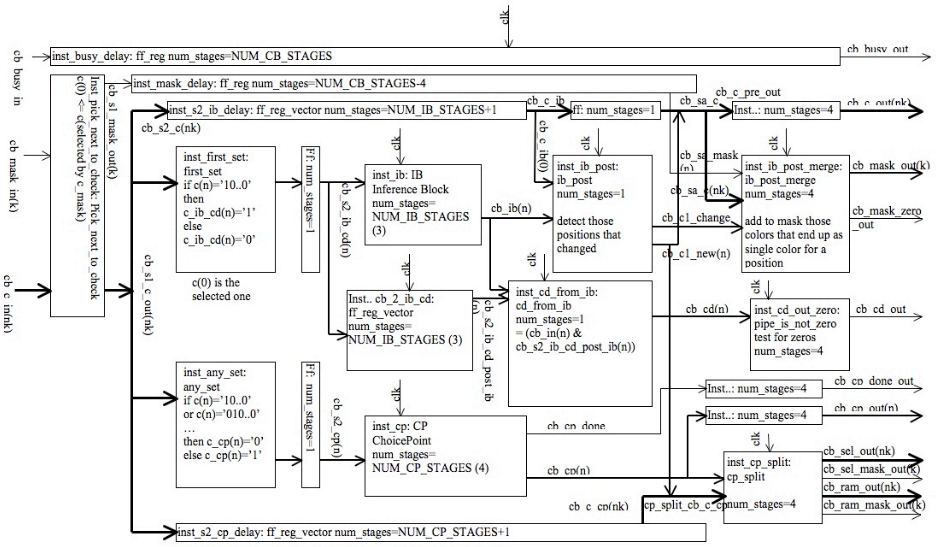


Figure 2: Combinatorial block for solver version 2



6.2. FPGA Solver Design: $K-$Inference Blocks Version

We later optimized the size of an inference block for $L - 1 \leq LUT_{inputs}$ (where LUT_{inputs} is equal to 6 for Virtex-5 and Virtex-6). The optimization involved trying to fit as many clauses on a single LUT. The algorithm does not find the minimum number of LUTs for a given formula, but significantly improves the optimization done by Xilinx ISE tool for the block, which finds inferences and contradictions. The optimization was most significant for smaller L (e.g., for $L = 3$ and $n = 128$, the number of LUTs was cut from roughly 7000 down to 2500 for the same Virtex-5 device[2]).

The reduction in size of an inference block allowed us to change the design and instantiate a separate inference block for each color. This reduction in size reduced the amount of support functions involved in deciding which color to check for inferences/contradictions next, thereby speeding up the design. It also allowed us to change the design to allow for off-diagonal computations (see results in Section 7).

6.3. PC-based Solver

Our method requires recovery of all satisfying assignments to the unassigned variables for initial assignments. The FPGA solvers used in our computation can only return 1 or 0 to indicate whether such an assignment exists, but not the actual solutions. The decision not to store the actual assignments was driven by concerns for FPGA space and added complexity. Hence we developed a PC-based solver to find all satisfying assignments given an initial assignment and a set of unassigned variables. Similar to the FPGA solver for $n = 200$, there are 181 unassigned variables, 90 placed before and 91 after a given pattern. This solver only processes the 23 assignments that the FPGA solver flagged as having a satisfiable assignment for $n = 200$ and collected all solutions.

7. Regression Testing and Comments on Previously Found Van der Waerden Numbers

Using our techniques we recomputed all known van der Waerden numbers listed in [2]. They were all verified and in one case corrected. For results see Tables 3 through 7. The recalculations were done as an additional integrity check of the solver. Furthermore, for each number we added information about the number of "extreme" colorings (number of satisfying assignments for $n - 1$ variables for each van der Waerden number n), and about the runtime on an FPGA at 100MHz.

During the regression testing, except where specifically noted, no preprocessing was done and run times could be greatly improved if, for example, simple pruning of redundant branches was applied.

We also included the time it took to build the solvers (synthesis and place-and-route times), which showed some of the drawbacks of using FPGAs for the computation of van der Waerden numbers. We also included statistics of the number of LUTs and slices occupied on a Virtex-6 FPGA chip. This information points out

[2]Xilinx ISE 10.1

how much area is occupied by the solver and points to possible scaling issues for larger van der Waerden numbers as another drawback of FPGAs.

Furthermore we utilized Cryptol to define formal models for many parts of the FPGA solvers and using the equivalence checking tool included with Cryptol verified many blocks of the solver.

Table 6 shows our new additions.

$W(K; L_1, L_2, \dots L_K)$	n	# of $n-1$	Runtime (secs)[3]	Synth./P&R (CPU time secs)[4]	LUTs/Slices
$W(2;3)$ [10]	9	6	< 10	13/256	968/300
$W(2;3,4)$ [10]	18	2	< 10	19/249	1325/555
$W(2;3,5)$ [10]	22	14	< 10	21/2940	1634/500
$W(2;3,6)$ [10]	32	12	< 10	30/295	2288/1109
$W(2;3,7)$ [10]	46	8	< 10	40/435	4065/1547
$W(2;3,8)$ [7]	58	2	< 10	53/658	4469/1927
$W(2;3,9)$ [7]	77	2	< 10	84/490	6208/3063
$W(2;3,10)$ [7]	97	16	< 10	135/850	8163/3952
$W(2;3,11)$ [14]	114	30	< 10	188/3994	10336/4152
$W(2;3,12)$ [14]	135	1	< 10	267/1488	14120/6136
$W(2;3,13)$ [14]	160	24	< 10	424/2700	18363/7193
$W(2;3,14)$ [12]	186	4	< 10	669/1894	25001/10199
$W(2;3,15)$ [12]	218	2	54	1101/2678	31938/12266
$W(2;3,16)$ [12]	238	38	311	1402/3151	36132/13920
$W(2;3,17)$ [4]	279	1	3075	2436/6508	47123/16987
$W(2;3,18)$ [4]	312	144	23533	3402/8210	57846/20930
$W(2;3,19)$ [6]	349	458	257086	5392/8303	68019/25624
$W(2;4)$ [10]	35	28	< 10	30/523	2405/1179
$W(2;4,5)$ [10]	55	65	< 10	49/1047	4652/1937
$W(2;4,6)$ [7]	73	2	< 10	87/663	6892/2996
$W(2;4,7)$ [8]	109	2	< 10	336/1044	15955/5681
$W(2;4,8)$ [12]	146	19	104	440/1863	16468/5973
$W(2;4,9)$ [5]	309	3	52330	5151/6836	63150/22524
$W(2;5)$ [18]	178	193624	< 10	708/938	31123/11670
$W(2;5,6)$ [12]	206	720	509	2360/1177	40383/16035
$W(2;6)$ [13]	1132	3552	276 hrs[5]	390/2906	107786/33799

Table 3: Known van der Waerden numbers for $K = 2$

[3]Running on 100MHz Xilinx ML605(Virtex 6) unless specified otherwise. The interface between FPGA and PC adds less than 10s overhead.

[4]Running on 2.66GHz Intel Core2 Duo with 6GB RAM.

[5]On 5x Xilinx ML605 at 200MHz each running 4 solvers for $n = 240$.

$W(K; L_1, L_2, \ldots L_K)$	n	# of $n-1$	Runtime (secs)[3]	Synth./P&R (CPU time secs)[4]	LUTs/Slices
$W(3; 2, 3, 3)$ [9]	14	2	< 10	16/178	1190/421
$W(3; 2, 3, 4)$ [9]	21	8	< 10	21/192	1607/682
$W(3; 2, 3, 5)$ [9]	32	2	< 10	32/240	2472/1171
$W(3; 2, 3, 6)$ [9]	40	20	< 10	33/266	3308/1427
$W(3; 2, 3, 7)$ [14]	55	36	< 10	56/316	5673/2316
$W(3; 2, 3, 8)$ [12]	72	14	< 10	82/468	6313/3255
$W(3; 2, 3, 9)$ [3]	90	6	< 10	123/555	8210/4079
$W(3; 2, 3, 10)$ [3]	108	2	< 10	172/6152	10878/3923
$W(3; 2, 3, 11)$ [3]	129	14	43	254/2553	13187/5542
$W(3; 2, 3, 12)$ [3]	150	2	148	366/1844	18058/7338
$W(3; 2, 3, 13)$ [3]	171	14	2295	532/2773	21921/8276
$W(3; 2, 4, 4)$ [9]	40	2	< 10	28/265	3216/1389
$W(3; 2, 4, 5)$ [9]	71	14	< 10	87/1282	7283/2635
$W(3; 2, 4, 6)$ [14]	83	12	< 10	127/530	9092/3802
$W(3; 2, 4, 7)$ [12]	119	14	337	464/1003	19238/6487
$W(3; 2, 5, 5)$ [3]	180	16444	25	760/1018	33039/13053
$W(3; 3)$ [10]	27	48	< 10	25/635	2168/860
$W(3; 3, 3, 4)$ [7]	51	16	< 10	45/955	4920/2393
$W(3; 3, 3, 5)$ [14]	80	84	21	112/946	9704/4774
$W(3; 3, 3, 6)$ [5]	107	180	1153	237/800	15935/6438
$W(3; 3, 4, 4)$ [14]	89	8	183	125/1029	11775/5239
$W(4; 2, 2, 3, 3)$ [9]	17	84	< 10	19/210	1494/537
$W(4; 2, 2, 3, 4)$ [9]	25	8	< 10	27/210	2085/858
$W(4; 2, 2, 3, 5)$ [9]	43	2	< 10	39/3727	3998/1363
$W(4; 2, 2, 3, 6)$ [14]	48	56	< 10	44/330	4483/2072
$W(4; 2, 2, 3, 7)$ [14]	65	10	< 10	83/582	7809/3169
$W(4; 2, 2, 3, 8)$ [3]	83	4	< 10	112/917	8094/3751
$W(4; 2, 2, 3, 9)$ [3]	99	16	51	156/580	10399/5369
$W(4; 2, 2, 3, 10)$ [3]	119	8	443	244/2630	13095/5093
$W(4; 2, 2, 3, 11)$ [16]	141	8	7015	336/2598	17625/6681
$W(4; 2, 2, 4, 4)$ [9]	53	16	< 10	43/687	4952/2242
$W(4; 2, 2, 4, 5)$ [3]	75	4	< 10	103/558	8207/3811
$W(4; 2, 2, 4, 6)$ [3]	93	16	280	177/754	12079/4251
$W(4; 2, 3, 3, 3)$ [9]	40	54	< 10	31/244	3635/1796
$W(4; 2, 3, 3, 4)$ [14]	60	16	< 10	63/494	6463/3352
$W(4; 2, 3, 3, 5)$ [3]	86	160	1157	138/1759	12099/5477
$W(4; 3)$ [7]	76	1440	152	73/4216	10378/5551

Table 4: Known van der Waerden numbers for $K = 3$ and $K = 4$

$W(K; L_1, L_2, \dots L_K)$	n	# of $n-1$	Run. (secs)[3]	Synt./P&R (CPUtime secs)[4]	LUTs/Slices
$W(5; 2, 2, 2, 3, 3)$ [14]	20	60	< 10	23/914	1838/581
$W(5; 2, 2, 2, 3, 4)$ [3]	29	24	< 10	27/223	2567/1099
$W(5; 2, 2, 2, 3, 5)$ [3]	44	132	< 10	44/702	4181/1933
$W(5; 2, 2, 2, 3, 6)$ [3]	56	12	18	61/339	5708/2725
$W(5; 2, 2, 2, 3, 7)$ [3]	72	24	39	110/448	9468/4325
$W(5; 2, 2, 2, 3, 8)$ [3]	88	12	471	134/2073	10203/4417
$W(5; 2, 2, 2, 4, 4)$ [3]	54	240	17	48/403	5363/2713
$W(5; 2, 2, 2, 4, 5)$ [3]	79	408	139	124/530	9256/3992
$W(5; 2, 2, 3, 3, 3)$ [14]	41	504	< 10	35/1329	4146/1472
$W(5; 2, 2, 3, 3, 4)$ [3]	63	80	136	73/444	7312/3683
$W(6; 2, 2, 2, 2, 3, 3)$ [3]	21	4560	< 10	28/288	2083/771
$W(6; 2, 2, 2, 2, 3, 4)$ [3]	33	144	< 10	33/262	3177/1342
$W(6; 2, 2, 2, 2, 3, 5)$ [3]	50	288	12	55/409	5334/2413
$W(6; 2, 2, 2, 2, 3, 6)$ [3]	60	96	117	74/486	6834/2980
$W(6; 2, 2, 2, 2, 4, 4)$ [3]	56	192	44	58/328	6118/2914
$W(6; 2, 2, 2, 3, 3, 3)$ [3]	42	7128	14	38/358	4472/2067
$W(7; 2, 2, 2, 2, 2, 3, 3)$ [3]	24	240	< 10	26/243	2582/997
$W(7; 2, 2, 2, 2, 2, 3, 4)$ [3]	36	3720	10	41/272	3866/1529
$W(7; 2, 2, 2, 2, 2, 3, 5)$ [5]	55	1920	286	72/383	6470/2969
$W(8; 2, 2, 2, 2, 2, 2, 3, 3)$ [3]	25	93600	10	29/286	2958/1012
$W(8; 2, 2, 2, 2, 2, 2, 3, 4)$ [5]	40	1440	265	49/338	4699/2001
$W(9; 2, 2, 2, 2, 2, 2, 2, 3, 3)$ [3]	28	131040	129	35/594	3489/1399

Table 5: Known van der Waerden numbers for $K = 5 \dots 9$

$W(K; L_1, L_2, \dots L_K)$	n	# of $n-1$	Runtime (secs)[3]	Synth./P&R (CPU time secs)[4]	LUTs/Slices
$W(3; 4, 4, 4)$	293	468	852 hrs[6]	391/8433	119412/34576
$W(3; 2, 3, 14)$[7]	202	1	26951	866/1553	30268/12489
$W(3; 2, 4, 8)$	157	24	159795	540/2512	21886/7925
$W(3; 2, 5, 6)$	246	4	562677	4625/1782	55528/19878
$W(4; 2, 2, 4, 7)$	143	132	119721	750/1150	26775/9154
$W(5; 2, 2, 2, 3, 9)$	107	48	6487	205/1042	12569/5616
$W(5; 2, 2, 2, 4, 6)$	101	12	46201	237/839	14808/5321

Table 6: Previously unknown van der Waerden numbers

[6]On 5x Xilinx ML605 at 150MHz each running 4 solvers for $n = 160$.

[7]Corrected earlier result found in literature.

8. Conclusion

We have demonstrated that $W(3,4)$ equals 293 using a problem-specific preprocessing that significantly cut down the search space, an efficient FPGA solver which took 3 months to run, a PC-based solver together with postprocessing to recover the search space, and an exhaustive expansion of the resulting patterns to obtain the complete set of extreme (i.e., of maximal length) colorings of length 292. We verified that those colorings do not extend further and hence established that $W(3,4) = 293$. Using these techniques we also established a number of previously unknown off-diagonal van der Waerden numbers. The specifically crafted SAT solver running on FPGAs, as well as cutting down on the search space via preprocessing, were the key components to accomplishing the task in a reasonable amount of time. The drawback of using FPGAs is the complexity of developing and running a solver. The synthesis and P&R time in Tables 3 through 6 show that a significant amount of time must be spent just to get the solver built, adding onto other challenges such as debugging, power consumption, heat dissipation, and so forth. Computational proofs almost entirely depend not only on the quality of coding, but also on the reliability of the computational infrastructure.

In addition to computing new van der Waerden numbers, we verified all currently known van der Waerden numbers. Moreover, in each case we computed the extreme colorings associated with these numbers, i.e., colorings without monochromatic a.p.s of the appropriate length for one less variable than the actual associated van der Waerden number. The extreme colorings are a critical component when implementing regression tests. Last but not least, we added several new upper bounds for van der Waerden numbers.

Acknowledgments This work has been supported by Cincinnati Childrens Hospital Medical Center, and in previous years by NSF Grants 9871345 and 0521189. The author would like to thank Jerry Paul for his contributions and help with this paper. The author also would like to thank William Gasarch for his review and helpful comments on a previous draft of this paper.

References

[1] Cryptol. `http://corp.galois.com/cryptol/`. Accessed: May 30, 2012.

[2] Ahmed, T. Van der Waerden numbers. `http://users.encs.concordia.ca/~ta_ahmed/vdw.html`. Accessed: May 30, 2012.

[3] Ahmed, T. Some new Van der Waerden numbers and some Van der Waerden-type numbers. *Integers* **9** (2009), 65–76.

[4] Ahmed, T. Two new van der Waerden numbers: $w(2;3,17)$ and $w(2;3,18)$. *Integers* **10** (2010), 369–377.

[5] Ahmed, T. On computation of exact Van der Waerden numbers. *Integers* **11** (2011), 1–7.

[6] Ahmed, T., Kullmann, O., and Snevily, H. On the van der Waerden numbers $w(2;3,t)$. *arXiv.org math.CO* (2011).

[7] Beeler, M. Some new van der Waerden numbers. *Discrete Mathematics* **28** (1979), 135–146.

[8] Beeler, M. D. A new Van der Waerden number. *Discrete Applied Mathematics* **6** (1983), 207.

[9] Brown, T. Some new van der Waerden numbers (preliminary report). *Notices American Math. Society* **21** (1974), A–432.

[10] Chvátal, V. *Some unknown van der Waerden numbers.* Combinatorial Structures and their Applications, 1970.

[11] Gowers, W. T. A new proof of Szemerédi's theorem. *GAFA* **11** (2001), 465–588.

[12] Kouril, M. A backtracking framework for beowulf clusters with an extension to multi-cluster computation and sat benchmark problem implementation. *Dissertation* (2006), 153.

[13] Kouril, M., and Paul, J. L. The van der Waerden number W (2, 6) is 1132. *Experimental Mathematics* **17**, 1 (2008), 53–61.

[14] Landman, B., Robertson, A., and Culver, C. Some new exact van der Waerden numbers. *INTEGERS* **5**, 2 (2005), A10.

[15] Rabung, J. Some progression-free partitions constructed using Folkman's method. *Canad. Math. Bull.* **22** (1979), 87–91.

[16] Schweitzer, P. Problems of unknown complexity: graph isomorphism and Ramsey theoretic numbers. *Dissertation zur Erlangung des Grades des Doktors der Naturwissenschaften* (2009).

[17] Shelah, S. Primitive recursive bounds for van der Waerden numbers. *J. Amer. Math. Soc.* **1** (1988), 683–697.

[18] Stevens, R. Computer-generated van der Waerden partitions. *Math. Computation* **32** (1978), 635–636.

[19] Van der Waerden, B. *Beweis einer baudetschen vermutung* **15** (1927) Nieuw Arch. Wisk.