Einschätzung Tool Chain - Xilinx Blockset + Matlab / Simulink

Kategorie	Problem	Beschreibung	Funktion	Handling	Skalierung	Kommentar
Arbeits-technik	Grafische Eingabe keine wirkliche Unterstützung	Matlab unterstützt mit dem Blockset nicht wirklich die Generation von VHDL sondern bietet nur die Möglichkeit, dies grafisch zu erledigen. Grundsätzliche Themen des timing und der Verschaltung bleiben unverändert bestehen. Das Ganze wird allerdings umständlicher, naturgemäß aufwändiger und es entsteht - anders, als man dies vermuten könnte - auch keineswegs automatisch mehr Übersicht.	Nur auf den ersten Blick "einfacher"	zusätzlicher Aufwand	einfaches Skalieren eines Designs erschwert	Grundsätzlich hinter-fragenswert
Arbeits-technik	Enge Kopplung der Darstellung und der Realisation u.U. kontra-produktiv	Der direkte Zusammenhang zwischen Grafik und Realisation führt dazu, dass alle Einzelheiten erst peinlichst genau spezifiziert werden müssen, damit das Design komplett funktionieren kann. Damit ist es a) nicht möglich, ein Design teilweise laufen zu lassen, während man es an einem anderen Teil weiterbaut, b) schwierig ein Zwischenergebnis zu bilden oder zu liefern, da erst Fehlstellen im Design ausgeheilt werden müssen. Es ist auch schwierig, komplexe Verzweigungen in einem Design mit minimalem Aufwand zu spezifizieren, oder durch eine andere Person, als den Designer selbst spezifizieren zu lassen		starr und pragmatisch	u.U. erschwert	Ansichtssache, wirkst sich je nach Aufgabe verschieden aus
Arbeits-technik	Teamwork stark erschwert	Grafische Tools erschweren generell das Team work. Grafische Sourcefiles lassen sind schwer bis gar nicht gleichzeitig bearbeiten oder pflegen. Damit wird z.B. auch die Verwaltung teilkonsistenter Designs und die Rückspeisung von Informationen schwieriger und aufwändiger.	Echtes Teamwork unmöglich	Erschwernis	Erschwernis, Mehraufwand	nicht überall einsetzbar
Arbeits-technik	Paralleles Arbeiten verhindert	Während einer laufenden Simulation oder Synthese kann nicht am Design weitergearbeitet werden. Selbst kleine grafische Änderungen werden teilweise nicht angenommen. Es müsste erst eine manuelle Kopie des Desings erzeugt-, mit MATLAB geladen und gestartet werden, um Simulieren UND Arbeiten zu können. Dabei wäre es wichtig, bereits während einer Simulation die ersten Ergebnisse in Änderungen einfließen zu lassen.	starke Einschränkung	Erschwernis, Mehraufwand	gar nicht dran zu denken	Mehr Freizeit für den Designer :-)
Arbeits-technik	Derivat-Verwaltung erschwert	Grafische Tools bieten durchaus grundsätzlich die Möglichkeit der Erzeugung und Verwaltung von Derivaten und haben hier sogar Vorteile gegenüber dem manuellen handling - sofern geeignete Projektverwaltung implementiert ist. Die grafischen Sourcefiles / Editor bieten hier aber - anders, als z.B. der HDL-Design - keine Unterstützung an. Damit wird z.B. die Entwicklung und Verwaltung teilkonsistenter Derivate durch / mit Kollegen schwieriger bis unmöglich.	Funktion nicht vorhanden	Erschwernis, Mehraufwand	gar nicht dran zu denken	Zurück in die Steinzeit der EDA
Arbeits-technik	Versionierung erschwert	Grafische Tools ohne zusätzlich aufgesetzte Dateiverwaltung erschweren generell die Generation und Verwaltung von Softwareversionen. Grafische Sourcefiles sind schwerer bis gar nicht automatisch zu versionieren und im Zuge von Quervergleichen auf Änderungen hin untersuchbar. In Matlab/Simulink fehlt eine entsprechende Funktion völlig - obwohl sie grundsätzlich implementierbar wäre und angesichts der heute bestehenden Anforderungen an Design und Dokumentation auch dringend notwendig ist.	Funktion nicht vorhanden	Erschwernis, Mehraufwand	gar nicht dran zu denken	Ein k.o. - Kriterium überall dort, wo Version management betrieben wird
Konzept	Records nicht nutzbar	Die in VHDL häufig verwendete Funktion der Records ist in Simulink nicht verfügbar. Damit werden designs unübersichtlich, es besteht ein großer Mehraufwand bei Portierung und eine Skalierung ist unmöglich.	nicht vorhanden	zusätzlicher Aufwand	einfaches Skalieren eines Design erschwert	Verhindert sicheres Handling großer Designs

Einschätzung Tool Chain - Xilinx Blockset + Matlab / Simulink

Konzept	Generics kaum nutzbar	Die in VHDL standardmäßig verwendete Funktion der Generics ist in Simulink nicht konsistent nutzbar. Dadurch werden designs statisch, es besteht ein Mehraufwand bei der Portierung und eine Skalierung wird sehr viel aufwändiger.	nicht vorhanden	zusätzlicher Aufwand	Durchgängiges Skalieren eines Design unmöglich	Eine vollkommen unakzeptable und massive Einschränkung !
Konzept	Counter haben indirekten Takt	Counter laufen ohne echten Takt. Sie müssen über eine virtuelle Periode parametriert werden, was bei krummen Frequenzen zu Diskrepanzen im Laufverhalten führen kann und es ferner nötig macht, den Counter jedesmals anzufassen, wenn die Taktfrequenz geändert wird.	Problem	Hindernis	automatisch Skalierung wird dadurch unterbrochen	ein überdenkenswertes Konzept
Konzept	Konstanten müssen manuell parametriert werden	Die Dimension von Variablen / Konstanten muss z.B. bei Komparatoren manuell angepasst werden, damit das Design simuliert, anstatt dass sie automatisch von MATLAB berechnet werden, was deterministisch möglich ist. Durch diese ausdrückliche Festlegung wiederum wird bei einer durchgängigen Änderung von Signalen ein erneutes Anfassen des blocks nötig.	mögliche Fehlerquelle	Erschwernis, Mehraufwand	automatisch Skalierung wird dadurch unterbrochen	vollkommen unnötig und widersinnig
Konzept	Daten-IO- müssen manuell parametriert werden	Die virtuellen Ports sowie die realen Schnittstellen zu Core-Elementen müssen bezüglich der Dateninterpretation manuell parametriert werden. Die im Design erforderliche Genauigkeit wird nicht von MATLAB durch das design durchgetrieben, sondern muss wie in VHDL auch punktuell definiert und verifiziert werden. Dabei ist Simulink aber nicht durchgängig: Einige Blöcke verarbeiten die Interpretation, andere nicht. Dies führt dazu, dass künstliche und z.T. unsinnige Interpretationen entstehen - bzw. aufgesetzt werden müssen, um Module überhaupt aneinander anschließen und verbinden zu können. Viele eigentlich simple Aktionen erfordern so sehr viele zusätzliche Eingaben, die in VHDL gar nicht benötigt würden. Die Designeingabe dauert länger und wird sehr schnell unübersichtlich.	Problem	zusätzlicher Aufwand, die Funktion der "Schaltung" wird weniger gut erkennbar, statt evidenter	Im Falle manueller Interpretationen wird Skalierung unterbrochen	Der Designer muss Simulink beim Rechnen helfen, statt umgekehrt. Das gesamte Konzept wird auf den Kopf gestellt!
Konzept	Pipelining muss per Hand erzeugt werden	Um Signale, welche aus unterschiedlichen Zeitebenen stammen mit einander verrechnen zu können, muss das durch die blocksets entstandene Delay manuell korrigiert werden, obwohl es aus den Cores bekannt ist, sein müsste. Dieser Schritt wäre leicht automatisierbar. Durch die manuelle Einstellung der Delays müssen Änderungen unter Umständen kettenförmig durch das Design durchgetrieben werden. In VHDL erfolgt eine derartige Anpassung durch Zeilenverschieben in getaktete Prozesse oder parametrierte Synthese von Delay FFs.	mögliche Fehlerquelle	zusätzlicher Design-Aufwand	nicht automatisch erzeugt und auch nicht durch Berechung skalierbar	Dies wäre das EIGENTLICHE Ziel der Übersetzung von MATLAB in VHDL
Konzept	Darstellung der Simulink-Werte ungeschickt	Die simulierten Werte werden in Simulink-Scopes nicht kontinuierlich dargestellt. Ein repaint nach verlorenem Windows-Fokus führt zu einer Ausschnittdarstellung. Erst nach dem Ende der Simulation ist das komplette Bild verfügbar. Die Bedienung der Scroll - und Zoomfunktionen ist mangelhaft	Ent-schleunigte Entwicklung	umständliche Bedienbarkeit		Angeblich soll es doch einfacher sein, als mit ModelSIM :-)
Konzept	Darstellung Wavescope-Werte ungeschickt	Werte werden im WAVE SCOPE nicht kontinuierlich dargestellt, sondern erst nachdem die Simulation durchgelaufen ist oder abgebrochen wurde. Die Bedienung der Scroll - und Zoomfunktionen ist eine Katastrophe	Ent-schleunigte Entwicklung	Sehr schlechte Bedienbarkeit		WAVE-SCOPE ist die übelste Komponente insgesamt!!
Konzept	Einfacher Counter erfordert Speziallizenz	Einfache binäre counter sind nicht nutzbar, da die Benutzung eines anderen Counters als 16 Bit eine spezielle "fixed point Lizenz" erfordert - die nicht verfügbar ist. Ein simples Durchtakten eines einfachen RAMs erfordert daher z.B. einen aufwändigen Zeichenkonstrukt, statt einer einzigen VHDL.-Zeile.	Funktion nicht verfügbar	workaround nötig		ein Witz!

Einschätzung Tool Chain - Xilinx Blockset + Matlab / Simulink

Realisation	Latenz in Cores verhindert Simulation	Aus einigen Cores kommen infolge von pipelining und Latenz zu Beginn einer Simulation naturgemäß unbekannte Werte. Diese werden von Simulink unterschiedlich interpretiert und gehandelt. Teilweise ist das tool nicht in der Lage, damit umzugehen und simuliert nicht, weil nachgeschaltete Einheiten keine vernünftigen Werte bekommen. Dies führt bereits beim check der Simulation zu einem Fehler und der Unmöglichkeit, zu simulieren. Damit tut die Simulation nicht, was sie eigentlich sollte: Die Information über Existenz sowie die Dauer dieser Latenz präzise nach Außen darstellen. Es müssen daher zusätzliche Komponenten nachgeschaltet werden, die zum startup den "U"-Datenstrom unterdrücken, damit die Schaltung simulieren kann, auch wenn dies real nicht von Nöten wäre.	Problem	zusätzlicher Aufwand	Unterbrochen	Typisches Problem solcher tools. Die Simulation sollte einfach laufen und "unknown" zeigen.
Realisation	Blockmodule komplex und versteckt parametriert	Viele Funktionen wie Addierer, Multiplizieren oder allgemeine mathematische Blöcke müssen sehr aufwändig parametriert werden, um simple Funktionen zu erzielen. So muss z.B. die allgemeine "math-Funktion" als EXP, SART o.ä. spezifiziert werden. Die genaue Funktion ist ihnen jedoch im grafischen Design oft nicht anzusehen. Daher müssen zusätzliche Kommentare eingefügt werden, um die Funktion zu beschreiben. Der Anspruch, das Design durch Grafik lesbarer zu machen, wird völlig konterkariert.	mögliche Fehlerquelle	zusätzlicher Aufwand	Unterbrochen	Die Übersicht und Lesbarkeit des Designs wird damit nicht besser, sondern schlechter!
Realisation	embedded VHDL wrapper umständlich und statisch	Um eingebettetes VHDL im Design nutzen zu können, muss es umständlich eingefasst werden. Der dazu zu erstellende Wrapper wird nicht immer automatisch erzeugt, hat teilweise Fehler und bedarf oft der Nacharbeit. Die Portdeklaration ist bezüglich der Vektoren nicht konsistent (einmal std_logic, ein anderes mal std_logic_vector), bezieht sich zudem auf veraltete Synopsis-Bibliotheken und ist nicht ausreichend sprachtolerant. (VHDL-Standard > 2002). Problem: Sobald z.B. die Portbreite im Design geändert wird, muss der Wrapper manuell angefasst werden. Unterkomponenten müssen per Hand angegeben werden, statt sie dem design zu entnehmen, generics werden unvollständig und falsch gehandhabt.	Einschränkung		Unterbrochen	Es ist nicht einsehbar, dass ein Tool, welches vorgibt, aus einfacher Mathematik VHDL erzeugen zu können, hier nicht ohne Hilfen durch den User klarkommt
Realisation	Mathematisches Abbild und VHDL Core nicht konsistent	Das von Simulink während der Simulation auf dem PC genutzte Modell mancher Blocksets stimmt nicht bitgenau mit dem überein, was hinterher in VHDL oder als CORE-Parameter realisiert wird. Dies führt sowohl zu scheinbar funktionierenden Designs die, später versagen, wie auch zu scheinbar nicht funktionierenden Designs, die lauffähig sind.	Problem	fatal error!		dies konterkariert die Vorgehensweise, M/S zu nutzen in vollkommenster Weise
Realisation	BitGen-Timingannahme nicht aussagefähig	Wird, wie in den meisten Fällen, kein vollständiges VHDL-Design angelegt, sondern nur eine Art drop-in des algorithmischen Teils erzeugt, welches in einem größeren Design eingefasst werden soll, wie z.B. mehrere Instanzen einer Recheneinheit, so versagt die vom SysGen erzeugte Timingvorhersage mangels ausreichender Isolation (IO-FFs zu den Ports) von einem potenziellen Restdesign. Das spätere Delay wird automatisch immer überschätzt.	Funktion nutzlos	zusätzlicher Synthese-Aufwand		Eine recht peinliche Schwäche. Gerade ein FPGA Hersteller sollte um diesen Effekt wissen.

Einschätzung Tool Chain - Xilinx Blockset + Matlab / Simulink

Realisation	Pipelining-Annahme nicht aussagefähig	Die Funktion "check pipelining", die anhand der Zielfrequenz die nötigen FFs nach einer Unit abschätzen will, arbeitet unzuverlässig. Das Delay wird in Einzelfällen unterschätzt. Zwar wird das lokale Element möglicherweise richtig eingeschätzt, in Verbindung mit Restdesign um das Simulink-Design herum, treten aber timing-Probleme auf.	Funktion nutzlos	zusätzlicher Synthese-Aufwand		ließe sich mit etwas Hirnschmalz durchaus beheben
Realisation	LoLevel Timing und Signale sehr schlecht beobachtbar	Um Delays und Timingaspekte präzise einstellen zu können, ist es erforderlich, Single auf Taktebene zu verfolgen. Das in Simulink angebotene Tool "WaveScope" arbeitet konzeptionell ungeschickt, hat eine Reihe von Bedienschwächen, die das Anzeigen verhindern und kommt überdies mit extrem vielen bugs: Werte werden gar nicht, oder nicht kontinuierlich angezeigt, Signale verschwinden, sie lassen sich infolge eines Programmierfehlers im popup-Window nach einer Weile nicht mehr skalieren oder sie ändern scheinbar ihre Namen. So werden falsche und verwirrende Informationen erzeugt, die es in Einzelfällen verunmöglichen, die korrekte Funktion der Signale zu prüfen.	Funktion weitgehend wertlos, teilweise fehlerhaft	zusätzlicher Verifikations-Aufwand	mehrdimensionale Signale nicht anzeigbar	Das Tool ist eine Zumutung und gelinde gesagt, eine Frechheit. Etwas anfängerhafteres habe ich selten gesehen.
Realisation	Port-numerierung nicht konsistent	Die Nummerierung der Ports, die die Reihenfolge der Pins in Symbolen bei Subsystemen und Co-Simulationsblöcken definiert, wird nicht konsistent abgebildet. Dies führt zu vollkommen unnötigem Mehraufwand bei der Generation von CO-Simulations-Designs und Alternativen und erschwert Änderungen und Updates		sehr viel zusätzlicher Zeichen-Aufwand		typisches Xilinx Chaos!
Realisation	Bitlevel Operationen unnötig kompliziert	Bei Zusammenfügen von Bitvektoren muss deren mathematische Wertinterpretationen manuell auf "neutral" gestellt- also die Information zerstört werden, um sie aneinanderbinden zu können, obwohl z.B. Bitvektoren grundsätzlich nicht notwendigerweise Zahlen darstellen, folglich keine diesbezügliche Interpretation besitzen (müssen). Das manuelle und ausdrückliche Zerstören der Information ist also im Fall unspezifizierter Werte (z.B. wenn aus einer Zahl ein Wert und gleichzeitig ein Takttiming abgeleitet wird) entweder unnötig oder redundant, da die Informationsverfälschung implizit bereits durch die Bitoperation erfolgt. Ein grundsätzliches "unknown" am Ausgang des Concat würde völlig reichen. In den Fällen, in denen hingegen spezifizierte Werte entstehen und die Interpretation eindeutig ist, wie z.B. bei einem Concat aus zwei tatsächlich mathematischen Werten, ist diese vorherige Zerstörung komplett unsinnig, da hinterher wieder eine händische Reinterpretation aufgesetzt werden muss, die aber volllkommen eindeutig ist, also durch MATLAB leistbar wäre.	weniger Übersicht, mangelnde Funktion, große Fehlerquelle	viel zusätzlicher und unnötiger Zeichen-Aufwand, stark verschlechtere Übersicht durch mehr Symbole	Eine System-Skalierung durch simplen Aufrechterhalt von Verknüpfungen aufgrund ausdrücklicher Interpretationen unterwegs unnötig verhindert!	Diese "Notwendigkeit" ist unsinnig sowie hinderlich und aus konzeptioneller Sicht irrational: Bitoperationen brauchen keine mathematische Interpretation. Dort hingegen, wo sie vorliegt und nutzbar wäre, wird sie irrigerweise ignoriert.
Realisation	Speicher-Operationen unnötig kompliziert	Bei der Realisation von Ram-Anschlüssen muss die mathematische Wertinterpretationen manuell auf "neutral" gestellt werden. Durch das Zerstören der Information ist es unmöglich, den Wert aus dem (dual ported) RAM beim Auslesen zu reinterpretieren. Bei dem simplen Auslesen aus einem RAM, in das vorher ein gleich breiter Vektor eingeschrieben wurde, ist diese Zerstörung komplett unsinnig, da wieder händisch reinterpretiert werden muss, was durch MATLAB leistbar wäre. Wo also in VHDL ein simples Zusammenfügen oder Anschalten eines Vektors gereicht hätte, muss die Stimmigkeit jedes Mal künstlich (wieder-) hergestellt werden.	weniger Übersicht, mangelnde Funktion, mögliche Fehlerquelle	viel zusätzlicher und unnötiger Zeichen-Aufwand, stark verschlechtere Übersicht durch mehr Symbole	Skalierung durch simples Fort-schreiben der Interpretation der Werte unmöglich, obwohl durchaus leistbar	Siehe oben! Was soll aus einem Ram denn bitte rauskommen, wenn nicht das, was man reinschreibt?

Einschätzung Tool Chain - Xilinx Blockset + Matlab / Simulink

Realisation	Parameter verändern sich	Parameter verändern sich oder werden nicht ordnungsgemäß gespeichert. Ein Hinweis auf Inkonsistenz sind fehlerhafte Grafikdarstellungen, die beim öffnen eines kürzlich geänderte Designs hochkommen: Hier sitzen Linien und Objekte an anderen Stellen, als ursprünglich platziert. Siehe auch den Effekt bei Wavescope.	fatal!			So ziemlich die größte Katastrophe!
Realisation	Fehler in VHDL wird mit falscher Zeilennummer angegeben	Gibt es bei der Übersetzung / Einbindung von VHDL einen Fehler, so wird die Zeilennummer des übersetzten Codes angegeben. Dieser enthält in einem großen file u.a. auch die Quell-VHDL-files. Damit wird eine Zeilennummer in der Größenordung von Tausenden ausgegeben, die nicht direkt auf die fehlerhafte Zeile im Code hindeutet.	eingeschränkter Nutzen	zusätzlicher Suchaufwand		Peinlicher Denkfehler!
Realisation	Hinzufügen funktioneller Pins verändert bestehendes Design	Beim nachträglichen Hinzufügen eines Ports an ein blockset Element, z.B. ein reset an einen counter, der zuvor nur ein enable hatte, wird der Reset zum Pin 1 des Elements, was dazu führt, dass der vormalige enable Pin in der Luft hängt, während das Signal, das diesen Pin im design ansteuert, nun fälschlicherweise den Reset bedient. Die Funktion Reset, kann also nicht einfach hinzugefügt, oder weggenommen werden, wie in VHDL, dabei sollte aus logischer sicht das Hinzufügen einer Funktion oder eines Elements ein Design erweitern, nicht aber bestehende Funktionen zerstören.	Funktion schlecht Nutzbar	unnötiger zusätzlicher Umbau-Aufwand	dynamisches Addieren / Nutzen von Funktionen (Derivat-Erzeugung) dadurch unmöglich	Welch ein Unsinn! Auch dieses kleine Detail zeigt, wie wenig Xilinx logisch denken kann.
Realisation	Speicherung von Simulations-Ergebnissen klappt nicht zuverlässig	Wenn eine Simulation aufgrund mangelnder Verfügbarkeit von Speicher abbricht, sind alle bis dahin ermittelten Ergebnisse verloren. Die Simulation muss zeitlich verkürzt und neu gestartet werden.		Mehraufwand		sehr nervig!
Realisation	Hinweise führen in die Irre	Hilfen sind nicht konsistent. Oftmals verweisen Fehlermeldungen auf Hilfethemen, die so nicht existieren, bzw. die in den boxen angegebenen Lösungen sind nicht anwendbar, da sie sich offenbar auf alte Versionen beziehen - sich die Menüpunkte und Anordnung in neueren Versionen längst geändert hat.	Funktion schlecht Nutzbar	zusätzlicher Suchaufwand		Raten ist angesagt
Realisation	Fehler-Meldungen nicht aussagefähig	Die während der Übersetzung gebrachten Fehlermeldungen sind häufig nicht schlüssig. Es werden Folgefehler gebracht, die nicht Ursache des Problems sind und z.B. Scheinfehler angemahnt, die infolge des Abbruchs des Compilationsvorgang durch den Compiler selber entstehen, z.B. unerledigte Variablenprüfungen.	Funktion schlecht Nutzbar	zusätzlicher Suchaufwand		Absolut Xilinx-typisch.