TR-O-0115

20

BSCMAアダプティブアレーアンテナディジタル信号処理部の開発

田中　豊久

1996. 3.15

ATR光電波通信研究所

目次

第１章　はじめに

昨今、世の中は移動通信ばやりである。電車の中、会議中、レストランで携帯端末の呼出音が鳴る。浸透の速度は目覚ましいものがある。人々は一度移動通信の便利さに気が付けばもう、元に戻れないかもしれない。この勢いで市場が需要が伸びれば将来的に周波数が欠乏状態になることは簡単に予測できる。この問題を解決するためには、新たな周波数の開拓、周波数の繰り返し利用、多重化技術の利用による回線の確保などが検討、研究されている。また、将来的には音声通信にとどまらず、画像通信、B-ISDNに代表される広帯域データ通信などの要求が高まってくるであろう。

一方、電子機器分野では、20年程度前からディジタル化の波が押し寄せて来ている。通信も例外ではなく、既にディジタル通信はサービスが開始されている。今後もディジタル化された通信は益々発展して行くものと予想される。

ATRでは設立当時から将来の高機能移動通信用アンテナとして、ディジタル信号処理を利用したDigital Beam Forming Antennaを提案、研究して来た。このアンテナはアンテナ部に円環パッチアンテナとリングアンテナを2層構造にして組み合わせたセルフダイプレクシングアンテナを用い、さらにそれをアクティブアレーアンテナ構成にしている。アナログ部も集積化が期待できるMMICによって構成も可能である。最後に複雑な信号処理を引き受けるディジタル部に並列処理ディジタル信号処理装置(DSP)で構成される、各種の技術の組み合わさった統合化アンテナである。当初ディジタル信号処理部は汎用DSPチップの搭載したボードを複数枚使用して構築されたDSPシステムを用いて研究が行われていた。汎用ボードを組み合わせたシステムでは、複雑な並列処理と相互のデータのやり取りを制御する事や動作速度の限界があり、また汎用システムであるがゆえに筐体も非常に大型にならざるを得なかった。ここで、より小型化、複雑な処理の実現を目指したDSP部のASIC化に白羽の矢が立った。最終的な目標はすべてのDBFアンテナの処理が1チップに集積化されたASICの完成であるが、1チップASICは作成にコストと汎用性を犠牲にする事を要求する。そこで設計がASICと同じ手法が用いられ、設計自体もASICに転用可能なFPGA(Field Programmable Gate Array)を用いたDSPを構築する事になった。FPGAを用いる利点として、ユーザサイトで設計開発が可能な事、ASIC化へのハード規模などの情報が得られる事が挙げられる。逆に、欠点としてはそのプログラミングが可能なアーキテクチャにより動作速度がASICと比べて劣っている。しかしながら実験システムではデータレートを低く抑える事により、その欠点をカバーすることができる。このようなATRでの歴史的背景のなか、FPGAを用いたDSPボードにより、DBFアンテナの実現性を検証する事を目的としたDSP開発が行われた。本研究レポートでは学術論文等では、取り扱わなかった、設計のノウハウを含めてDBFアンテナが作れるようにまた、既存のシステムが理解できるように記述した。DBF技術を用いたFFTマルチビーム生成部、ビームセレクタ部、位相補正部までを参考文献[1-1]に、CMA(Constant modulus Algorithm)を指導原理としたアダプティブプロセッサについてを本テクニカルレポートに、ビームスペースでアレーアンテナで最大比合成を行なうDBFセルフビームステアリング用プロセッサについては参考文献[1-2]で述べる。

第２章　Beam Space CMA Adaptive Array Antenna

移動体通信に於いては、地域的に同じ周波数を共有するための周波数の有効利用や、通信の省電力化、干渉波の除去による通信品質の向上などの必要性が論じられている。これらの課題を改善する為の手段として、空間的な指向性を実現出来るマルチビームアンテナ、あるいは干渉波を除去する為のアダプティブ信号処理などが提案されてきている[2-1]-[2-7]。ディジタルビームフォーミング(DBF)アンテナはこれらの手段を実現する方法として有望であり[2-1]、急速なデバイスの高速化および大規模化に伴い注目されつつある。このDBFアンテナを用いれば、マルチビームフォーミング、受信信号の自動追尾、干渉波除去、同相合成によるSNRの改善などの機能を得る事ができる。またこれにより、基地局及び移動局の出力の省電力化や周波数の有効利用などが期待出来る。DBFアンテナは受信信号の高速処理や複雑な信号処理の為、ディジタル信号処理部のASIC(Application Specific Integrated Circuit)化が不可避となる。ASICとして書き換え可能なFPGA(Field Programmable Gate Array) を用いた実時間でのマルチビームフォーミング機能、自動追尾機能、およびCMA (Constant Modulus Algorithm)を指導原理としたアダプティブ機能を備えたビームスペースCMA(BSCMA)アダプティブアンテナ用ディジタル信号処理ASICの設計について述べる。

BSCMAアダプティブアンテナは、個々の素子で受信するSNRの低い信号を、マルチビームによってビーム合成する事により改善する。その出力の大きいビームを用いてアダプティブ処理を行う事により、素子アンテナに対応させてCMAアダプティブ処理する場合に比べ演算時間、ハードウェア規模を削減できる特徴を持つ[2-7]。ここでは、試作したBSCMAアダプティブアンテナの構成について説明し、ついで各部の動作原理について述べる。

第１節BSCMAアダプティブアンテナシステムの概要

アンテナ部は図2-1.に示すように、素子アンテナは4×4の四角形に配置し、16素子の2層構造のマイクロストリップセルフダイプレクシングアンテナを用いている。FFT演算を採用する為、隣接素子アンテナは$\lambda/2$ (λ:搬送波の波長)の等間隔に並べられている。搬送波周波数はL帯(1.545GHz)である。またデータレートは16kbpsである。図2.に受信部BSCMAアンテナの構成図を示す。試作システムはアンテナ部、周波数変換部、フィルター部、A/Dコンバータ部、ディジタル信号処理部からなっている。次に簡単に各ブロックの動作について説明する。各アンテナ素子ごとに受信された信号は周波数変換部(D/C)により、1.545GHzから32kHzのIF信号に変換される。続くフィルタ部にて高調波が除去され、次に32kHzのIF信号はサンプリング周波数128kHzでA/Dコンバータで8bitのディジタル信号に変換される。次に16素子からの受信データはディジタル信号処理部に渡される。ディジタル信号処理部では、16本のマルチビームが同時に形成され[2-3]、BSCMAアダプティブ処理用に最大4本のビームが振幅が大きいものから順に選択される。選択されたビームは、FFT演算時に位相がそれぞれ端の素子アンテナに基準が取られている。このため、ビームの切り替え時に位相のずれが生じる。したがってアンテナ中心への移相部では、位相中心をアンテナの中央部に移相する事によって、同じ入射角度におけるメインローブを持つ各ビームを同相にする演算を行う。次に移相されたビームはCMAアダプティブ部に渡され、アダプティブ処理が行われる。

表2-1. 開発システム概要

ITEM	SPECIFICATION
搬送波周波数	*1.545GHz (L帯)*
アンテナ素子数	*16素子（４×４）*
IF周波数	*32kHz*
A/Dコンバータ サンプリング周波数	*128kHz*
データレート	*16kbps*
変調方式	*π/4シフトQPSK*

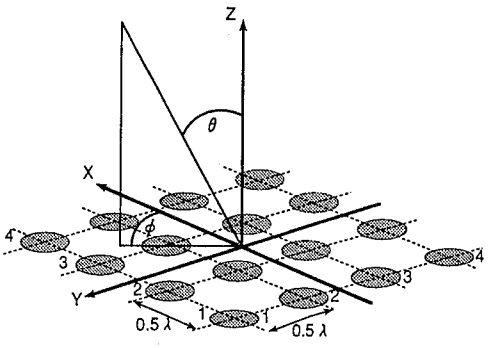


図2-1. アレーアンテナの配置図と座標系

第２節 アナログ部概要

図2-2.にDBFアンテナのブロック図を示す。アナログ部はアンテナ部、周波数変換部、フィルター部、A/Dコンバータ部からなる。各アンテナ素子ごとに受信された信号は周波数変換部(D/C)で、1.545GHzから32kHzのIF信号に変換され、続くフィルタ部で高調波が除去される。次にIF信号はサンプリング周波数128kHzでA/Dコンバータにより8bitのディジタル信号に変換された後、ディジタル信号処理部に渡される。ディジタル信号処理部に関する詳細は以降の章で機能毎に述べる事とする。

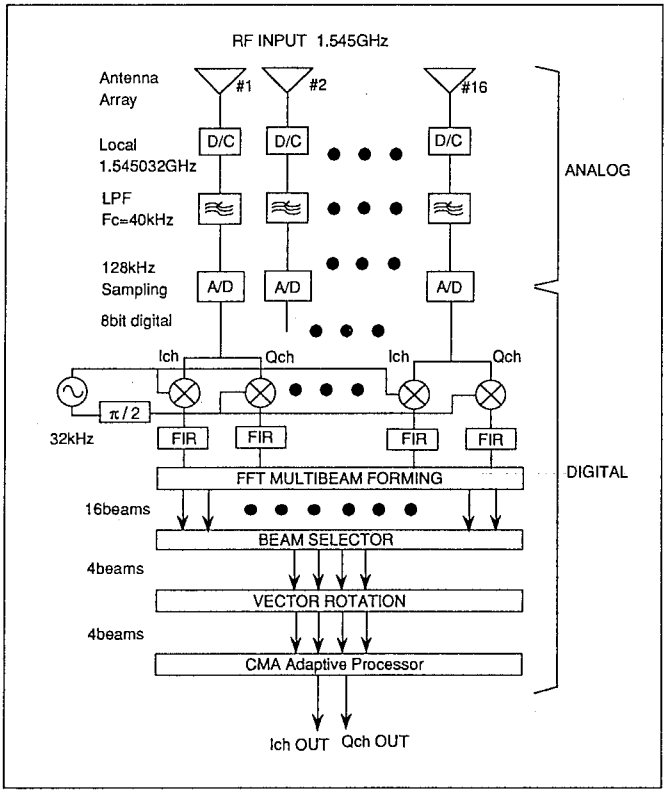


図2-2. DBFアンテナ構成図

第３章　BSCMA Adaptive Processor ASIC Implementation

第１節　BSCMAアダプティブアンテナの動作概要 [2-7]

ディジタル信号処理部では、図3-1.に示すように3段階のパイプライン処理によって各サンプル毎にリアルタイムにデータを出力する。3段階の処理はそれぞれ、マルチビームを合成するステージ、ビームを選択し移相を行うステージ、そしてCMAアダプティブステージである。マルチビームステージでは次の手順で信号処理される。まず素子アンテナの受信信号 S_{xy} は、

$$S_{xy}(t_l)=\cos(\omega_0 t_l+\phi_m+\theta_{xy}) \tag{3-1}$$

x=1,2,3,4 y=1,2,3,4

と表わされる。ここでx,yは素子アンテナの座標を表わし(図2-1.参照)、ω_0 はIF信号の角周波数、t_l はサンプル時間を表わす。ϕ_m はディジタルQPSKによる変調位相である。θ_{xy} は各アンテナ素子における受信位相の共通のローカル信号からのずれを表わす。まず入力データ式(3-1)は同相成分と直交成分に分ける為、32kHzの固定の位相のローカル信号を掛けられ、式2で表わされる準同期検波される。ローカル信号は4サンプルで1周期する信号で、0、$\pi/2$、π、$3\pi/2$のくり返しである。

$$\begin{bmatrix} i(x,y) \\ q(x,y) \end{bmatrix}=\begin{bmatrix} \cos(n) & 0 \\ 0 & -\sin(n) \end{bmatrix}\begin{bmatrix} S_{xy}(t_l) \\ S_{xy}(t_l) \end{bmatrix} \tag{3-2}$$

x=1,2,3,4　y=1,2,3,4　n=0, $\pi/2$, π, $3\pi/2$.....

次にIch、Qchのデータは式(3-3)に示す10タップのFIRディジタルフィルタを通り、準同期検波時に発生した高調波が取り除かれる。この演算は現在と過去9個の入力I,Qデータに係数を掛ける積和演算である。

$$\begin{bmatrix} i_f(x,y) \\ q_f(x,y) \end{bmatrix}=\sum_{n=0}^{9} h(n)\begin{bmatrix} i_{k-n}(x,y) \\ q_{k-n}(x,y) \end{bmatrix} \tag{3-3}$$

x=1,2,3,4　y=1,2,3,4

i_f 及び q_f はそれぞれフィルタリングされたベースバンド信号である。$i_k(x,y)$、$q_k(x,y)$ はそれぞれ現在のサンプル時のデータを表わす。次に各ベースバンド信号はビーム合成を行う為、FFT部に渡される。ここでFFTは2次元的に行われる。まず図1.のアンテナの配置と同様に考えれば、式(3-4)で示される様にX軸方向に沿って4素子ずつFFT演算される。

$$\begin{bmatrix} i'(x,y) \\ q'(x,y) \end{bmatrix}=\sum_{n=0}^{3}\begin{bmatrix} \cos(-nx\frac{\pi}{2}) & -\sin(-nx\frac{\pi}{2}) \\ \sin(-nx\frac{\pi}{2}) & \cos(-nx\frac{\pi}{2}) \end{bmatrix}\begin{bmatrix} i_f(n,y) \\ q_f(n,y) \end{bmatrix} \tag{3-4}$$

x=1,2,3,4　y=1,2,3,4

i'とq'はそれぞれ2次元FFTの中間結果を表わす。次にX軸方向に計算されたデータを用いて、式(3-5)に表わされるY軸方向に沿った２次元目のFFT演算が行われる。

$$\begin{bmatrix} I(X,Y) \\ Q(X,Y) \end{bmatrix}=\sum_{n=0}^{3}\begin{bmatrix} \cos(-ny\frac{\pi}{2}) & -\sin(-ny\frac{\pi}{2}) \\ \sin(-ny\frac{\pi}{2}) & \cos(-ny\frac{\pi}{2}) \end{bmatrix}\begin{bmatrix} i'(x,n) \\ q'(x,n) \end{bmatrix} \tag{3-5}$$

x=1,2,3,4　y=1,2,3,4　X=1,2,3,4　Y=1,2,3,4

この結果、同時に16本のビームのデータが得られる。$I(X,Y)$ および $Q(X,Y)$ はそれぞれ、合成されたビームを表わし、X、Yはビームの方向に対応している。合成されたビームは、次のサンプルの時に、ビーム選択とアンテナの中心部への移相のステージに渡される。このステージでは15個の振幅比較器を用い、最大4本のビームを選択する。ここでの選択方法は、まずトーナメント式に最も振幅の大きいビームを選びだし、次に選択されたビームを除外して再度比較を行い２番目のビームを選択し、同様に3番目、4番目のビームを選択する方法を用いている。この時、ある基準レベル以下のビームは選択しないようになっている。これは例えば図1で $\theta=0^\circ$、$\phi=0^\circ$ の時は、0° ビームのみで受かっており、その他のビームは直交しているのでノイズレベルにあり、入力ビームとしてCMAでの使用に適さないからであ

る。ビームが選択されると式3-6に示されるビーム毎の移相が行われる。

$$\begin{bmatrix} I'_{xy} \\ Q'_{xy} \end{bmatrix} = \begin{bmatrix} \cos(\phi_{xy}) & -\sin(\phi_{xy}) \\ \sin(\phi_{xy}) & \cos(\phi_{xy}) \end{bmatrix} \begin{bmatrix} I_{xy} \\ Q_{xy} \end{bmatrix} \tag{3-6}$$

$$\phi_{xy} = \exp(-j(\frac{3\pi(x-1)}{4}))\exp(-j(\frac{3\pi(y-1)}{4})),$$

x'=0,1,2,3　y'=0,1,2,3

ここでx'=0は図1のアンテナ配置のx=4であり、y'=0 は同様に y=4である。この意味は、x=4およびy=4の時 30° ビームであり、ビームとしてはx=1, y=1の時と対称の方向からであるので、x=4、y=4ではなくx=0、y=0として考慮しなければならないからである。次のサンプル時に、移相されたビームはCMAアダプティブステージに渡され、まず式3-7に示されるウェイトとの積和演算がなされる。

$$y_n = \sum_{m=1}^{4} W_m^{(n)} \cdot X_{mn} \tag{3-7}$$

nはn回目のサンプリングを表わし、X_{mn}は入力のビームを、$W_m^{(n)}$はm番目に選択されたビーム用のウェイトを表わしている。y_nは得られたCMAアダプティブステージの出力である。各々のウェイトベクターは式(3-8)に示される最急降下法により、サンプリングごとに更新される。

$$W_m^{(n+1)} = W_m^{(n)} - \mu X_{mn}^* y_n (|y_n|^2 - \sigma^2) \tag{3-8}$$

ここで、σは包絡線の目標値を表わし、μはステップ定数、X_{mn}^*は入力のビームの共役複素数を表わす。選択されたビームの組み合わせが変わる時、ウェイトベクターは初期化される。これらの演算の結果として、干渉波がある状況ではその方向にビームのヌルが作られ、マルチビームの谷間ではSNRを改善する様にビームが合成される。

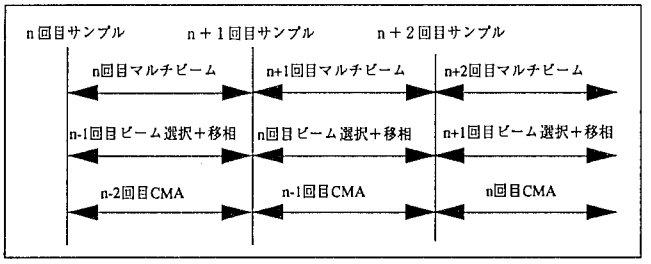


図3-1 ディジタル信号処理部でのパイプライン処理

表3-1 FFT演算のxy軸と各ビームの関係

Y\X	1	2	3	4
1	beam0	beam4	beam8	beam12
2	beam1	beam5	beam9	beam13
3	beam2	beam6	beam10	beam14
4	beam3	beam7	beam11	beam15

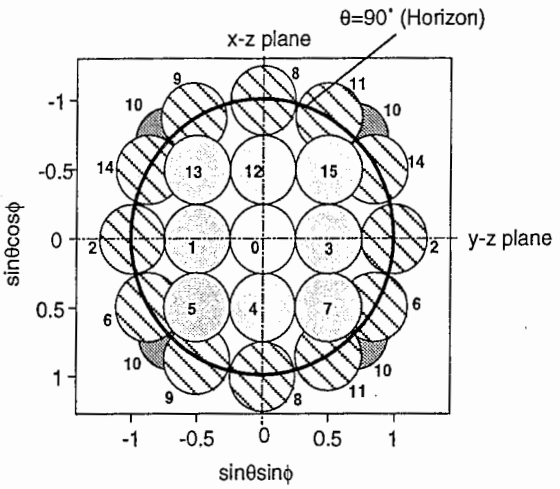


図3-2. ビームの方向とビーム番号

第2節 BSCMAアダプティブアンテナディジタル部概要 [3-1]-[3-3]

BSCMAアダプティブアンテナディジタル信号処理装置は2枚のボードで構成されている。ボードは、16個のA/Dコンバータを搭載したフロントエンドボードと10個のFPGAをDSP ASICとして搭載したDSPボードである。

BSCMAアダプティブアンテナ用ディジタル信号処理ASIC部は10個の書き換え可能なFPGAから構成されている。10個のFPGAのうち8個がマルチビーム生成部として使用されている。このステージでは、16個のプロセッサが用いられ、それぞれがA/Dコンバータからの出力を1チャンネルで1個ずつに対応する。1個のFPGAに2個のプロセッサが構築されている。位相中心への移相部とメインコントローラ、CPUインターフェイス部、クロック分配部が1個のFPGAに構築され、残りの1個のFPGAには最大4本のビームを選択する部分とCMA部が構築されている。表3-2にこれらの回路を構築するために必要となった等価ゲート数をまとめる。

表3-2. BSCMAアダプティブに必要となった等価ゲート数

機能	必要となったゲート数
マルチビーム生成部	106,125
ビーム選択部	2,889
位相補正部	6,107
CMAアダプティブ部 12bit	13,917
その他 (CPU I/F, etc.)	3,605
合計	132,643
(参考) CMAアダプティブ部 8bit	8,546

BSCMA用DSPの全体構成を図3-3.にまとめる。機能的には、マルチビーム生成ブロック、最大ビーム選択ブロック、位相中心移相ブロック、そしてCMAアダプティブブロックから成っている。これらは図3-1.に示すように各ステージでは、それぞれサンプル時間内に一連の演算を行い、次ステージに渡すパイプライン処理を行っている。

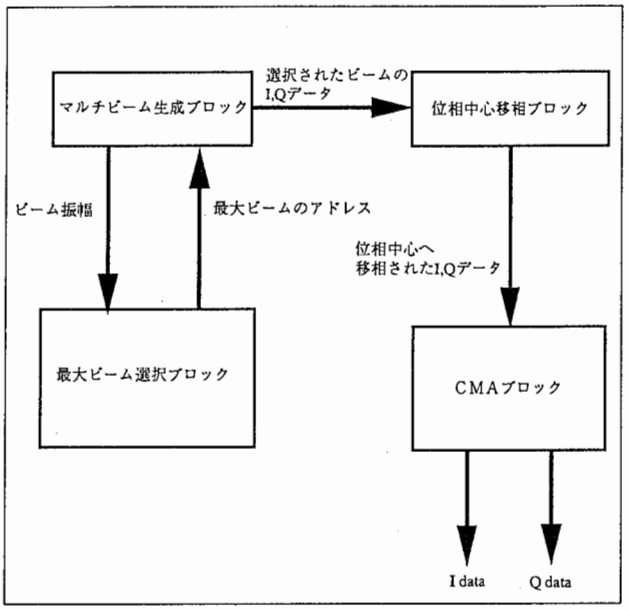


図3-3. BSCMAアダプティブDSPの全体構成図

第3節 CMAアダプティブ処理部の構成

第1項 Components

1. 12bit 固定小数点プロセッサ

CMAアダプティブ部での演算は12-bitの固定小数点精度で行われている。実現できるハードウェア規模に余裕がある場合、スケーリングなどに注意すれば、精度を上げる事も当然できる。12-bit精度は使用したFPGAのゲート数の制限により決まった。

1-1 マルチプライヤ部(Multiplier部)

Multiplier部は2つの12-bitデータを入力とする1の演算を行なう部分と24bitのfull adderから構成されている。DSP内では、特に断らない限り2の補数形式のデータを扱うものとする。

1. A × B = Carry値と演算値
2. Carry値＋演算値

このマルチプライヤ部の1は完全な組み合わせ回路で実現されている。FPGAの開発ツールにはこの部分のlibraryがないので、VHDLで記述しそれを論理合成ツールを用いて回路に変換し用いている。また、特に本FPGAの設計のプロセッサ部の動作速度はこの部分で決まっていると言っても良いと考えられる。これは、Accumulatorのように完全に最適化されたライブラリとして配置配線できないためである。2の部分はLibraryとして供給されている24bit Adderを用いた。

Scaling

DSPの固定小数点演算にはスケーリングは付き物であり、通常、積算の後は結果の上位ビットを用いる。この時に、データの精度を確保するため最適なスケーリングを行なう。例えば、12bit×12bitの演算の結果は24bitデータが得られる、この時に上位の11bit目から23bit目までを結果とするか、10bit目から22bit目までを

結果とするかを決定する事である。この部分をアダプティブに制御する方法もあるが簡単のため試作DSPでは固定のスケーリングファクタにした。

Overflow clip

入力信号が大い場合などに、ダイナミックレンジを越えるような計算結果が得られたり、その精度で表現できる数よりも大きくなってしまった場合、オーバーフローになる。オーバーフローになれば符号が反転しその数値も反転してしまう。これを避けるための方法として上位ビットと符号ビットをモニタしそれを越えた場合、最上値あるいは最下値にすればよいが、試作DSPでは簡単なゲートを組み合わせてオーバーフローのモニタを行なっている。

Rounding error

丸め誤差はスケーリングを行なう時に生じる。スケーリング時にただ上位ビットを次段に送った場合、2の補数形式ではマイナスにオフセットがかかってしまう。これを避けるために切り捨てるビットの最上位で0捨1入の処理を行なう。これを実現するテクニックとしては、

1. 切り捨て前のコンポーネントの出力に必要なビット数+1のアダーを配置し、最下位ビットに1を足す。
2. AccumulatorのLoad機能を利用する。

1-2 24 bit Accumulator部

積和演算の和の部分を行なうComponentである。Xilinxのlibraryの24bit Accumulatorをそのまま使用している。各演算グループの開始直前には必ずResetあるいは丸め誤差改善用のデータをAccumulatorにロードしておき使用する。ResetあるいはLoad信号はsequencerから制御される。

1-3 Pipelining

プロセッサ部は1 MCLKですべての処理がなされるのではなく、中間レジスタを用いて3段のパイプライン構成で処理される。パイプライン処理のタイミングを図3-4に示す。試作DSPでのシーケンスについて説明する。システム動作速度を上げる必要があれば、さらに中間レジスタを増やす方法もある。しかし、その分必要となるハードウェア規模も大きくなる。

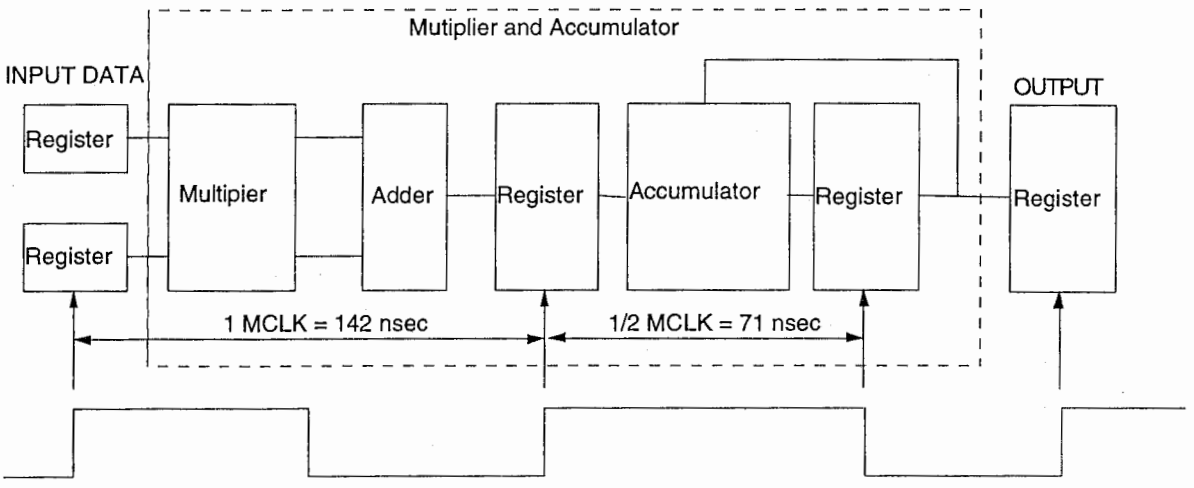


図3-4. プロセッサ部のパイプライン処理

1. MCLKの立上がりエッジでプロセッサ直前の入力レジスタに2つのデータがラッチされる。
2. 次のMCLKの立上がりまでの間に積算を完了させる。
3. 積算結果はMCLKの立上がりエッジで、中間レジスタにラッチされる。
4. その結果はAccumulatorの入力に渡され、Accumulator内のレジスタの数値と和減算が行なわれる。
5. 4の積和結果はMCLKの立ち下がりエッジでAccumulator内のレジスタにラッチされる。
以上のシーケンスで、1回の積和演算が1.5MCLKで完了する。パイプライン処理は演算の速度を向上させる事ができるが、中間レジスタによって回路規模の増大になり、また直前の演算結果を用いて次の演算を行なう処理などは待ち時間が生じてしまう欠点がある。逆に言えば、FIRフィルタのように連続して積和演算を行なうようなアルゴリズムは効率がよい。最終的な結果は5の演算完了後のMCLKの立上がりエッジで出力用のレジスタにラッチする。

2. 13 bit Subtractor

ウェイト更新時に前ウェイトと誤差分との差をとる時に用いられている。ハードウェア記述言語により記述されており、丸め誤差を考慮して設計されている。誤差分のデータはAccumulatorの出力ではなく、12bit マルチプライヤの出力レジスタから渡される。この時、12bitではなく1bit下位のデータを含めて受ける。そのデータが1の場合、誤差分データに1を足し、0の場合はそのまま前ウェイトとの減算を行なう。オーバフロー防止のため、ステータス出力も備えており、次段のレジスタはオーバフローのステータスを受けた場合、レジスタ出力を飽和データにする。(2047或は、-2048)

3. Output & intermidiate register

これは、Accumulatorからの出力を1時的に保持するためのレジスタである。出力レジスタにはCMA出力のIch、Qch用のレジスタ(cma_reg12)と中間レジスタの役割をするレジスタ(cma_reg12)がある。これらにはオーバーフロー防止用の回路が付加されている。その他のレジスタとして各入力ベクトルとCMA出力ベクトルの復素掛け算を行なった結果を保持するためのレジスタ(in_reg12)がある。
ラッチのトリガ信号は各sequencerから送り出される。トリガ信号の遅延量の影響を避けるため、レジスタのトリガイネーブル端子にトリガ信号は与えられ、トリガイネーブル状態でレジスタのクロック入力端子に繋がれたClock信号のエッジでラッチは行われる。

4. Multiplexer

入力切り替えのためのコンポーネントであり、制御信号は各sequencerで発生される。またMultiplexerの替わりにトライステートのバッファとバスを用いて入力切り替えはできる。試作DSPでは使用できるBUSの本数に制限があったので、BUSの必要な回路部分(CPU I/Fのデータバスなど)を優先して設計したため、プロセッサへの入力はMultiplexerを用いている。

5. Input register
プロセッサ直前のレジスタや、外部からのデータを保持するレジスタの事を指す。

6. Channel Register/Threshold Register

ビーム選択シーケンスにおいて、電力値の比較が行われるが、この時、ビーム番号と設定スレッシュレベルより大きいかという情報も同時に流される。これらを保持するためのレジスタである。

7. Wn RAM

アダプティブ処理のウェイトを保持するために全チャンネル用にRAMを用いた。初期バージョンでは、レ

ジスタを用いていたが、ビーム選択の順序が入れ代わるとウェイトとビームの関係も入れ代わってしまう構成であった。これをビームとウェイト値を対応させるためRAM方式に変更した。

8．Slave sequencer

CMAアダプティブ部のコントロールを行なうのはss_u9というシーケンサである。このシーケンサで取り扱う制御信号は、

1. CMAアダプティブ部の各種マルチプレクサの切り替え信号
2. 各レジスタへのラッチトリガ信号
3. ウェイト用RAMのライトイネーブル信号
4. プロセッサ部へのコントロール信号(加減算コントロール、リセットコントロール)

などである。具体的な信号名とコントロールタイミングについては、後述するTiming Chartの項で述べる。

第2項　CMAアダプティブプロセッシングのためのハードウェア構成

CMAアダプティブ部は図3-5.に示す構成になっている。ここでなされる演算はすべて単純な積和と加減算のみであるので、ハードウェアは非常に簡単な構成で実現可能である。CMAアダプティブ部は積和演算器、減算器、マルチプレクサ、レジスタ、そしてコントロール部で構成されている。

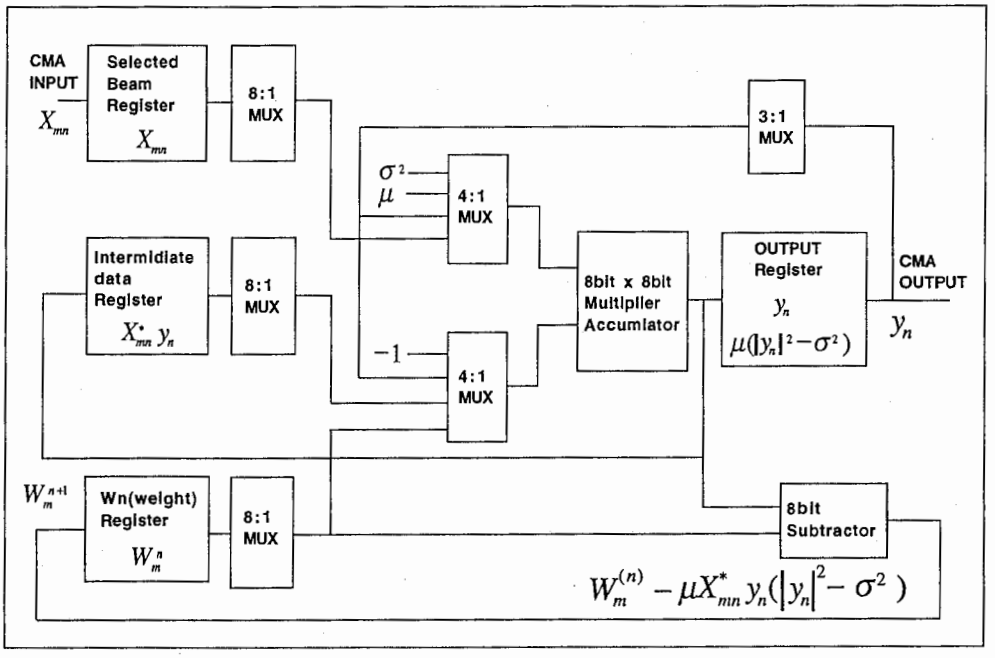


図3-5. CMAアダプティブ部の構成図

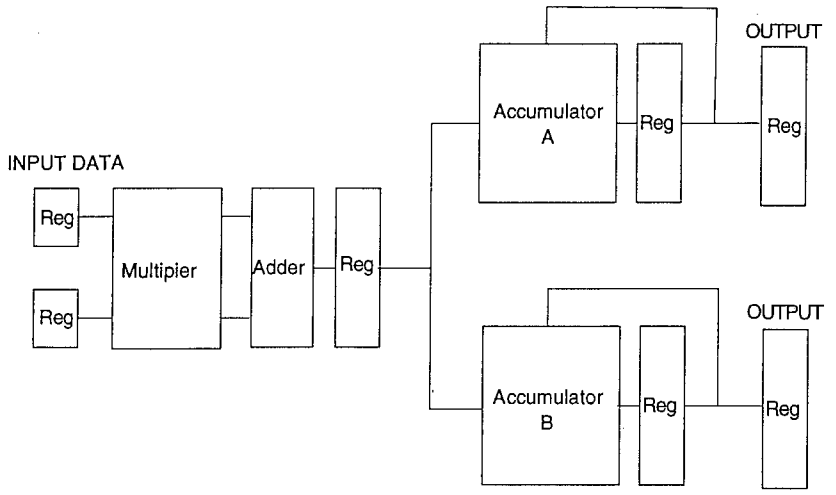


図3-6. CMAアダプティブ部のプロセッサ部の構成

図3-6はCMAアダプティブプロセッサ部の構成である。FPGAの動作速度が7.04MHz程度が限界であったため、1回のsampling間隔でかつ1つの演算器で最大55回の演算しかできない。また前項で述べたように、プロセッサ部ではパイプライン構成をとっている。従い、直前のデータを利用するためにはウェイト時間が発生する。また、丸め誤差改善のため、１つのシーケンスが終了する時に、次のシーケンスに向けて桁上げ用のデータをあらかじめAccumulatorのレジスタにロードするためのサイクルが必要となる。ここでいうシーケンスとは例えば、連続した積和演算、最終出力を求める入力ベクトルとアダプティブウェイトを掛け合わせ、積算していく事などを1シーケンスという。ここで、それらのウェイト時間を生じさせないため、Accumulatorを2個用い、マルチプライヤを共通に用いるという方法をとった。この方法ではウェイトの更新のための演算中、交互にAccumulatorを使用する事により、レジスタへのロードのウェイト時間は生じない。

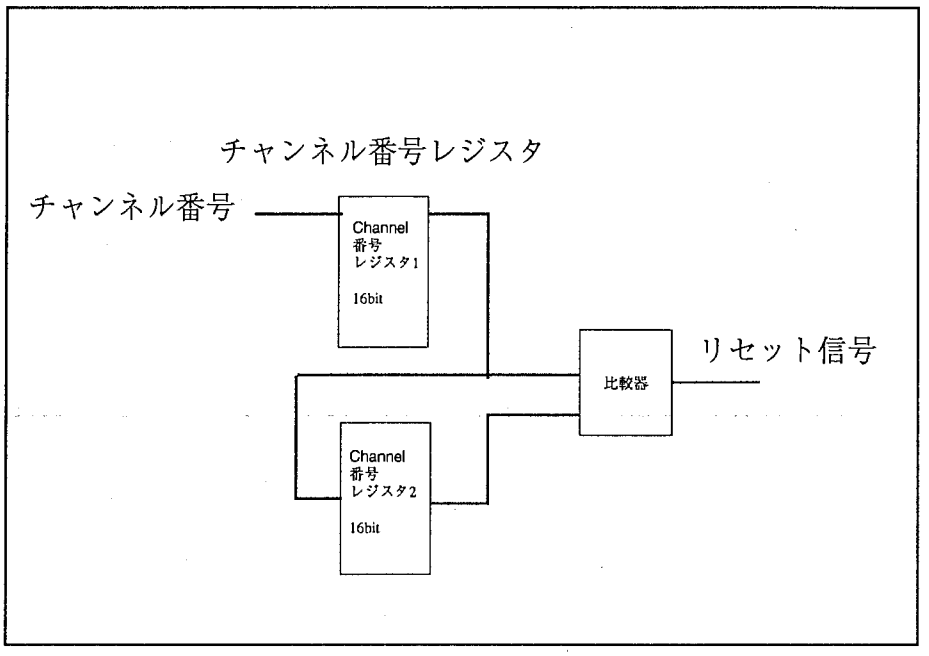


図3-7. ウェイトリセット信号発生部

入力ビームの組み合わせが変わった時にリセット信号を発生させる回路について図3-7に示す。この回路はビーム選択および移相のフェイズにて動作する。このブロックにおける動作は、まず選択されたビームのチャンネル番号をチャンネル番号レジスタ１に記憶する。このレジスタはnチャンネルのシステムの場合nビットのレジスタである。選択の動作が終わると、m本の選択の場合、チャンネル番号レジスタにはm-bitだ

け、1が書き込まれており、選択されたビームの組み合わせがわかる。チャンネル番号レジスタ２には前回に選択されたチャンネル番号が記憶されている。チャンネル番号レジスタ１とチャンネル番号レジスタ２を比較し、結果が異なっている場合にリセット信号を発生する。この信号によって次段のCMA部ではCMAの演算が行われる前に、重みが初期化される。また、設計CMAアダプティブプロセッサではこのリセットをマスクしてビームの組み合わせが変っても、ウェイトとリセットしないようにする事も可能にしている。実際にはアンテナを回転させ所望波にトラッキングさせる場合、通信としてビット誤り率を指標とすれば、リセットを行なわない方がよい結果が得られている。

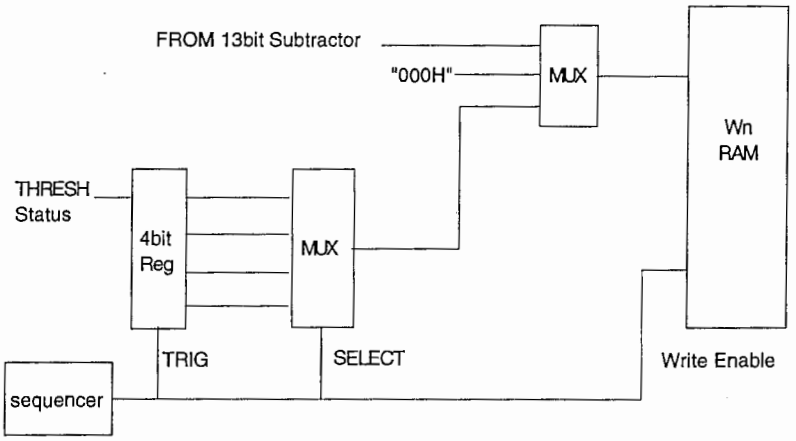


図3-6. Thresh levelの取り扱い

入力ベクトルが設定敷居値以下の場合の動作は、図3-6の回路で実現される。電力値比較のステージにおいて同時に設定敷居値との比較も行なっていると述べたが、その情報は4bitレジスタに保持される。次のウェイト更新時に設定敷居値以下のビームの場合、MUXで更新されたビームのウェイトの替わりに"000H"の方がWn RAMに対して出力されるようになる。

第３項　CMAアダプティブプロセッシングアルゴリズム概要とTiming Chart

CMAアダプティブの演算のフローを図3-7に示す。まず、選択されたビームベクターとウェイトベクターを積算しCMAアダプティブ部の出力を求める。実数部、虚数部の順序で演算を行なうが、2つのAccumulatorを用いたため、その間ウェイト時間(NOP)は生じていない。続いて、ウェイトベクターの更新の演算がなされる。まずy_nの電力値を求める計算を行なう。実数部と虚数部が2乗され加算される。そのy_n電力値から収束目標値σを差し引くのであるが、そのまま積和演算で行なう事を考え、ある適当な掛け算を行い、そのままAccumulatorで減算を行なうようにした。ここで得られた値にステップ定数μを掛けるのであるが、パイプラインで行なっているため、ここではウェイト時間が生じている。この時点で更新量を決める共通のスカラー量が得られた。次に各入力ベクトル値に応じた演算を行なうつまり、X^*_{mn}とy_nの復素掛け算を行なう。ここでも2つのAccumulatorを用いてウェイト時間が生じないようにしている。その復素掛け算の結果は中間レジスタ(in_reg12)に保持される。次に、各ウェイト毎に前ウェイトからの更新を行なうフェーズに入る。中間レジスタに保持されたベクトル値と先に得られたスカラー値を掛け合わせ、その結果はAccumulatorを用いずに直接13bit Subtractorに渡され、同時に13bit Subtractorに前ウェイトが出力されており、更新が行われる。この時12bit精度で表現できる最大、最小値はそれぞれ2047(7FFH)、-2048(800H)であるのでそれを越える場合はオーバーフローになり、符号が逆転してしまうので、オーバーフローのステータスを受けた時、次段のレジスタは飽和値を用いる事にした。最終の演算中ウェイト時間を持たせているが、ウェイト値を保持するRAMに書き込みを行なうために生じた。以上でCMAアダプティブ処理部の1sampleの演算が終了する。

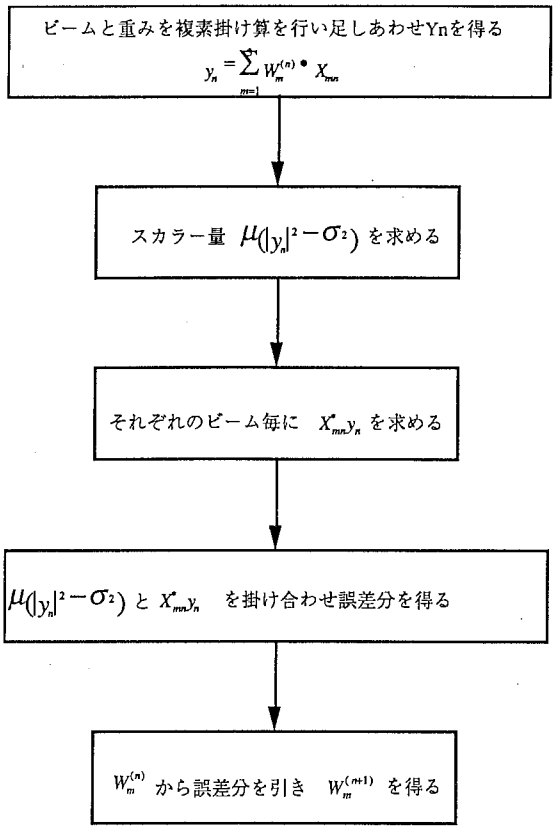


図3-7. CMAアダプティブ部の演算フロー

CMAアダプティブ処理部のTiming Chartを付録に添付する。Timing Chartでは各sequencerからの制御信号のTimingと各サンプリング周期の間に実行する演算の制御の為のフローを示している。Timing Chartに示されている制御信号について説明する。

u9のsseqによるコントロール

ROW_MUX: 入力ベクトルデータの出力切り替えコントロール
DLT_MUX: 中間レジスタに保持されたデータの出力切り替えコントロール
WNR_MUX ビーム番号を保持しているレジスタで、実数部の演算時の出力切り替えのコントロール用
WNI_MUX ビーム番号を保持しているレジスタで、虚数部の演算時の出力切り替えのコントロール用
A_MUX プロセッサへの入力切り替えコントロール
B_MUX プロセッサへの入力切り替えコントロール
OUT_MUX y_nと更新量を決めるスカラー値の出力切り替えコントロール
SUBREG_TRIG 前ウェイト値と更新値の減算後に保持するレジスタへのトリガ信号
ACC_EN Accumulatorへの丸め誤差補正用データのロードを行なう。(Accumulatorの初期化)
WN_RAM_WE ウェイトRAMの書き込みイネーブル信号
DLT_TRIG 中間レジスタに保持するためのトリガ信号
ROW_TRIG 入力レジスタに保持するためのトリガ信号
CHREG_TRIG ビーム番号保持用レジスタへのトリガ信号
OUT_TRIG y_nと更新量を決めるスカラー値を保持するレジスタへのトリガ信号
AS_CONT Accumulatorの演算で加算か減算を制御するための信号
WNRI_MUX ウェイト実数部用RAMか虚数部用RAMの出力切り替えコントロール
COMP_TRIG ビーム番号を保持するレジスタへのトリガ信号

第 4 章　むすび

ASICを用いたCMAアダプティブ処理部の設計・試作について述べた。試作は12bit固定小数点精度で構築したが、さらに高精度化をはかれば安定な動作と深いヌルビームフォーミングが可能であると思われる。またカスタムLSIを適応すればさらに大規模な構成にでき、また微細化技術による高速化も可能である。今後の発展として最急降下法での原理的な収束速度を改善するため別のアルゴリズムの実ハードウェアへの適応を検討課題としたい。

第 5 章　謝辞

日頃より御指導頂く、ATR光電波通信研究所 猪股英行社長に深く感謝いたします。また本研究を進めるにあたり正しい方向へお導き頂いた、唐沢好男室長、千葉　勇元主任研究員、三浦　龍主任研究員に深く感謝いたします。また、無線通信第一研究室の皆様に感謝いたします。

第 6 章　参考文献

[1-1] 田中豊久, "DBFマルチビームアンテナディジタル信号処理部の開発," ATRテクニカルレポート, TR-O-0114 (TR-O-0117、設計データソフト付き), Mar. 1996.

[1-2] 田中豊久, "DBFセルフビームステアリングアレーアンテナディジタル信号処理部の開発," ATRテクニカルレポート, TR-O-0116 (TR-O-0119、設計データソフト付き), Mar. 1996.

[2-1] K. Takao and K. Uchida, "Beamspace Partially adaptive antenna," IEE Proc., Pt. H, 6, pp.439-444 Dec. 1989.

[2-2] 藤元美俊、菊間信良、稲垣直樹、"マルカート法を用いたCMAアダプティブアレーの多重波抑制特性," 信学論(B-II)、Vol.J74-B-II, No.11, pp.599-607, Nov. (1991).

[2-3] T. Ohgane, T. Shimura, N. Matsuzawa and H. Sasaoka, "An implementation of a CMA adaptive array for high speed GMSK transmission in mobile communications," IEEE Trans. Veh. Technol., vol. 42, pp. 282-288, Aug. 1993.

[2-4] Y. Ogawa, Y. Nagashima and K. Itoh, "An Adaptive Antenna System for High-Speed Digital Mobile Communications," IEICE Trans. Commun., Vol. E75-B, No.5 May 1992.

[2-5]黒岩　登、河野隆二, "アダプティブアレーアンテナによる指向性ダイバーシチ受信の構成法," 信学論(B-II)、Vol.J73-B-II No.11, pp755-763, Nov. (1990).

[2-6] T. Ohgane, "Spectral Efficiency Improvement by Base Station Antenna Pattern Control for Land Mobile Cellular Systems," IEICE Trans. Commun., VOL E77-B, No. 5 May (1994).

[2-7] 千葉　勇、中條　渉、藤瀬　雅行: "ビームスペースCMAアダプティブアレーアンテナ,"信学論(B-II) Vol J77-B-II, No.3, pp.130-138, Mar. 1994.

[2-8] H. Steyskal, "Digital Beamforming Antennas," Microwave Journal, Jan. 1987, pp107-114.

[3-1] 大滝幸夫, "移動体衛星通信用DBFアンテナ信号処理部の構成とその特性," ATRテクニカルレポート, TR-O-0046, Jun. 1992.

[3-2] 田中豊久, 三浦　龍, 千葉　勇, 唐沢好男、"ASICを用いたDBFマルチビームアンテナの開発," 信学論(B-II), Vol.J78-B-II, no.9, pp602-610, Sep. 1995.

[3-3] T. Tanaka, R. Miura, I. Chiba, and Y. Karasawa, "An ASIC Implementation Scheme to Realize a Beam Space CMA Adaptive Array Antenna," IEICE Trans. commun., vol. E78-B, no.11, pp. 1467-1473, Nov. 1995.

第 7 章　付録

第 1 節　DSP ASIC Bd 回路図

第 1 項　u9

sheet1　Beam Selector

sheet2　CMA Adaptive processor MAIN Block

sheet3　FPGA I/O ビーム電力データ(u1、u2)

sheet4　FPGA I/O ビーム電力データ(u3、u4)

sheet5　FPGA I/O ビーム電力データ(u5、u6)

sheet6 FPGA I/O ビーム電力データ(u7、u8)
sheet7 FPGA I/O CMA出力
sheet8 FPGA I/O ビーム電力出力Control信号
sheet9 FPGA I/O CMA演算モニタ出力
sheet10 FPGA I/O Control信号、Beam最下位ビットアドレス、選択ビームアドレス
sheet11 FPGA I/O CPU I/F データバス
sheet12 CPU I/F(アドレスデコーダ、R/W、ライトパルス、リードイネーブル信号発生)

第 2 項 Components
mux82_1
mux84_1
ss_comp4
row_reg83
reg44
mux44_1
ss_u9p6
thr_reg
ff
proc12x12cma
mux12_8
reg12
ram16x12
mux122
mux124
fdrs12
FDRS
cma_reg12
in_reg12
fdce12
sub13of
x74_374
x74_37412
x74_3744

第 2 節 Misc Information

第 1 項 Timing Chart

第 3 節 VHDL Listing
第 1 項 ss_u9 VHDL Listing

-- Written by Toyohisa Tanaka @Radio Communications Department
-- Rev.A Thu, October, 5, 1995
--
-- Slave Sequencer for U9 Phase 6
--

```
-- Funtion
--
-- 1. ROW MUX(2:0) Control
-- 2. WN_MUX(2:0) Control     --> change
-- 3. DLT_MUX(2:0) Control    --> change
-- 4. A_MUX(1:0) Control      --> change
-- 5. B_MUX(1:0) Control      --> change
-- 6. ROW_TRIG(3:0) Control
-- 7. DLT_TRIG(7:0) Control
-- 8. WN_TRIG(7:0) Control    --> change
-- 9. OUT_TRIG(2:0) Control
-- 10. ACC_ENA, ACC_ENB, AS_CONT
-- 11. OUT_MUX(1:0) Control

-- a. SUB reg I,Q Trig
-- b. IN Trig EN for Channel registers
-- c. Wn MUX for I, Q
-- d. We for Wn RAM

-- MRST and START are active low signal in this module.
-- MCLK2 has been eliminated due to minimize clock skew.
-- OUT_EN and FIR_SEL are proceeded 1/2 MCLK, because of timing optimization.

library mgc_portable;
use mgc_portable.qsim_logic.all;

-- Slave Sequencer Entity Description

entity ss_u9p6 is
  port(
     MCLK, START, MRST: in qsim_state; -- eliminated MCLK2
       ROW_MUX, DLT_MUX: out qsim_state_vector(2 downto 0);
       WNR_MUX, WNI_MUX, A_MUX, B_MUX, OUT_MUX, SUBREG_TRIG, ACC_EN, WN_RAM_WE:
out qsim_state_vector(1 downto 0);
       DLT_TRIG: out qsim_state_vector(7 downto 0);
       ROW_TRIG, CHREG_TRIG: out qsim_state_vector(3 downto 0);
       OUT_TRIG: out qsim_state_vector(2 downto 0);
       AS_CONT, WNRI_MUX, COMP_TRIG: out qsim_state
   );
end ss_u9p6;

-- sseq_u9p6 Architecture Description

architecture rtl of ss_u9p6 is

-----------------------------------------------------------------
-- Signal declaration for STATE GENERATOR
-----------------------------------------------------------------
```

```
    signal start2: qsim_state := '1';
-------------------------------------------------------------
-- Signal declaration for MCLK sensitive signal
-------------------------------------------------------------
-------------------------------------------------------------
-- Signal declaration for MCLK2 sensitive signal
-------------------------------------------------------------

    signal count, count2, istate: integer range 0 to 63 :=0;
    signal mux_sel1, mux_sel3: qsim_state_vector(2 downto 0);
    signal mux_sela, mux_selb, mux_selo, mux_sel2, mux_sel4: qsim_state_vector(1 downto 0);
    signal trig2: qsim_state_vector(7 downto 0);
    signal trig4: qsim_state_vector(2 downto 0);
    signal trig1, trig5: qsim_state_vector(3 downto 0);
    signal en2, trig3, we1: qsim_state_vector(1 downto 0);
    signal en3, mux_sel5, trig6: qsim_state;

    type type2_states is (s00, s01);
    type type3_states is (s10, s11, s12);
    type type4_states is (s20, s21, s22, s23);
    type type5_states is (s30, s31, s32, s33, s34);
    type type8_states is (s0, s1, s2, s3, s4, s5, s6, s7);
    type type9_states is (s40, s41, s42, s43, s44, s45, s46, s47, s48);

    signal present_mux1_state  : type8_states := s0;
    signal next_mux1_state     : type8_states := s1;
    signal present_mux2_state  : type4_states := s20;
    signal next_mux2_state     : type4_states := s21;
    signal present_mux3_state  : type8_states := s0;
    signal next_mux3_state     : type8_states := s1;
    signal present_mux4_state  : type4_states := s20;
    signal next_mux4_state     : type4_states := s21;
    signal present_mux5_state  : type2_states := s00;
    signal next_mux5_state     : type2_states := s01;

    signal present_muxa_state  : type4_states := s20;
    signal next_muxa_state     : type4_states := s21;
    signal present_muxb_state  : type4_states := s20;
    signal next_muxb_state     : type4_states := s21;
    signal present_muxo_state  : type3_states := s10;
    signal next_muxo_state     : type3_states := s11;
    signal present_en2_state   : type3_states := s10;
    signal next_en2_state      : type3_states := s11;
    signal present_en3_state   : type2_states := s00;
    signal next_en3_state      : type2_states := s01;
    signal present_we1_state   : type3_states := s10;
    signal next_we1_state      : type3_states := s11;
```

```
        signal present_trig1_state : type5_states := s30;
        signal next_trig1_state    : type5_states := s31;
        signal present_trig2_state : type9_states := s40;
        signal next_trig2_state    : type9_states := s41;
        signal present_trig3_state : type3_states := s10;
        signal next_trig3_state    : type3_states := s11;
        signal present_trig4_state : type4_states := s20;
        signal next_trig4_state    : type4_states := s21;
        signal present_trig5_state : type5_states := s30;
        signal next_trig5_state    : type5_states := s31;
        signal present_trig6_state : type2_states := s00;
        signal next_trig6_state    : type2_states := s01;

begin
 START2_PROCESS:process(MRST, MCLK)
 begin
  if ( MRST = '1') then
      start2 <= '1';
  elsif ( MCLK'event and MCLK = '0' and MCLK'last_value = '1') then
      start2 <= START;
  end if;
 end process START2_PROCESS;

 INCREMENT_PROCESS:process(istate)
 begin
  count <= istate;
 end process INCREMENT_PROCESS;

 INIT_SEQ:process(START,MCLK) -- incremented by MCLK2 = count
 begin
    if ( START = '0' ) then -- Initializing local phase counters
       istate <= 0;
    elsif (MCLK'event and (MCLK = '0') and (MCLK'last_value = '1')) then
              istate <= count +1;
    end if;
 end process INIT_SEQ;

 INIT_SEQ2:process(start2,MCLK) -- incremented by MCLK = count2
 begin
    if ( start2 = '0' ) then
          count2 <= 0;
    elsif (MCLK'event and (MCLK = '1') and (MCLK'last_value = '0')) then
          count2 <=  count;
    end if;
 end process INIT_SEQ2;

-------------------------------------------------------------------------
-- MUX CONTROL
```

```
-------------------------------------------------------------------------

-------------------------------------------------------------------------
-- ROW_MUX CONTROL
-------------------------------------------------------------------------
 MUX1_STATE_DECODE:process(count)
 begin
 next_mux1_state <= s0;
 case count is
  when 1 | 10 | 24 | 26 => next_mux1_state <= s0;  -- **************************
  when 2 | 9 | 25 | 27 => next_mux1_state <= s1;  -- *************************
  when 3 | 12 | 28 | 30 => next_mux1_state <= s2;  -- **************************
  when 4 | 11 | 29 | 31 => next_mux1_state <= s3;  -- **************************
  when 5 | 14 | 32 | 34 => next_mux1_state <= s4;  -- **************************
  when 6 | 13 | 33 | 35 => next_mux1_state <= s5;  -- **************************
  when 7 | 16 | 36 | 38 => next_mux1_state <= s6;  -- **************************
  when 8 | 15 | 37 | 39 => next_mux1_state <= s7;  -- **************************
  when OTHERS => next_mux1_state <= s0;
 end case;
 end process MUX1_STATE_DECODE;

-------------------------------------------------------------------------
 MUX1_OUTPUT_DECODE:process(present_mux1_state)
 begin
  case present_mux1_state is
   when s0 => mux_sel1 <= "000";
   when s1 => mux_sel1 <= "001";
   when s2 => mux_sel1 <= "010";
   when s3 => mux_sel1 <= "011";
   when s4 => mux_sel1 <= "100";
   when s5 => mux_sel1 <= "101";
   when s6 => mux_sel1 <= "110";
   when s7 => mux_sel1 <= "111";
   when others => mux_sel1 <= "000"; -- Select A
  end case;
 end process MUX1_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- WNR_MUX CONTROL
-------------------------------------------------------------------------
 MUX2_STATE_DECODE:process(count)
 begin
 next_mux2_state <= s20;
 case count is
  when 3 | 4 | 11 | 12 | 44 | 45 | 46 => next_mux2_state <= s21;  -- *************
  when 5 | 6 | 13 | 14 | 47 | 48 | 49 => next_mux2_state <= s22;  -- *************
  when 7 | 8 | 15 | 16 | 50 | 51 | 52 => next_mux2_state <= s23;  -- *************
  when OTHERS => next_mux2_state <= s20;
```

```
 end case;
 end process MUX2_STATE_DECODE;

-------------------------------------------------------------------------
 MUX2_OUTPUT_DECODE:process(present_mux2_state)
 begin
  case present_mux2_state is
   when s21 => mux_sel2 <= "01";
   when s22 => mux_sel2 <= "10";
   when s23 => mux_sel2 <= "11";
   when others => mux_sel2 <= "00"; -- Select No. 1 channel
  end case;
 end process MUX2_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- WNI_MUX CONTROL
-------------------------------------------------------------------------
 MUX4_STATE_DECODE:process(count)
 begin
 next_mux4_state <= s20;
 case count is
   when 3|4|11|12|45|46|47 => next_mux4_state <= s21;  -- ************
   when 5|6|13|14|48|49|50 => next_mux4_state <= s22;  -- ************
   when 7|8|15|16|51|52|53 => next_mux4_state <= s23;  -- ************
   when OTHERS => next_mux4_state <= s20;
 end case;
 end process MUX4_STATE_DECODE;

-------------------------------------------------------------------------
 MUX4_OUTPUT_DECODE:process(present_mux4_state)
 begin
  case present_mux4_state is
   when s21 => mux_sel4 <= "01";
   when s22 => mux_sel4 <= "10";
   when s23 => mux_sel4 <= "11";
   when others => mux_sel4 <= "00"; -- Select No. 1 channel
  end case;
 end process MUX4_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- WNRI_MUX CONTROL
-------------------------------------------------------------------------
 MUX5_STATE_DECODE:process(count)
 begin
 next_mux5_state <= s00;
 case count is
   when 2|4|6|8|10|12|14|16|42|45|48|51 => next_mux5_state <= s01;  -- ************
   when OTHERS => next_mux5_state <= s00;
```

```
 end case;
 end process MUX5_STATE_DECODE;

 ---------------------------------------------------------------------------
 MUX5_OUTPUT_DECODE:process(present_mux5_state)
 begin
  case present_mux5_state is
   when s01 => mux_sel5 <= '1';
   when others => mux_sel5 <= '0'; -- Select No. 1 channel
  end case;
 end process MUX5_OUTPUT_DECODE;

 ---------------------------------------------------------------------------
-- WN_RAM_WE CONTROL
 ---------------------------------------------------------------------------
 WE1_STATE_DECODE:process(count)
 begin
 next_we1_state <= s10;
 case count is
   when 42 | 45 | 48 | 51 => next_we1_state <= s11;  -- *******
   when 43 | 46 | 49 | 52 => next_we1_state <= s12;  -- *******
   when OTHERS => next_we1_state <= s10;
 end case;
 end process WE1_STATE_DECODE;

 ---------------------------------------------------------------------------
 WE1_OUTPUT_DECODE:process(present_we1_state)
 begin
  case present_we1_state is
   when s11 => we1 <= "01";
   when s12 => we1 <= "10";
   when others => we1 <= "00"; -- NOT WE
  end case;
 end process WE1_OUTPUT_DECODE;

 ---------------------------------------------------------------------------
-- DLT_MUX CONTROL
 ---------------------------------------------------------------------------
 MUX3_STATE_DECODE:process(count)
 begin
 next_mux3_state <= s0;
 case count is
   when 40 => next_mux3_state <= s0;  -- ****************************
   when 41 => next_mux3_state <= s1;  -- ****************************
   when 43 => next_mux3_state <= s2;  -- ****************************
   when 44 => next_mux3_state <= s3;  -- ****************************
   when 46 => next_mux3_state <= s4;  -- ****************************
   when 47 => next_mux3_state <= s5;  -- ****************************
```

```
  when 49 => next_mux3_state <= s6;  -- ***************************
  when 50 => next_mux3_state <= s7;  -- ***************************
  when OTHERS => next_mux3_state <= s0;
 end case;
 end process MUX3_STATE_DECODE;

-------------------------------------------------------------------------
 MUX3_OUTPUT_DECODE:process(present_mux3_state)
 begin
  case present_mux3_state is
   when s0 => mux_sel3 <= "000";
   when s1 => mux_sel3 <= "001";
   when s2 => mux_sel3 <= "010";
   when s3 => mux_sel3 <= "011";
   when s4 => mux_sel3 <= "100";
   when s5 => mux_sel3 <= "101";
   when s6 => mux_sel3 <= "110";
   when s7 => mux_sel3 <= "111";
   when others => mux_sel3 <= "000"; -- Select A
  end case;
 end process MUX3_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- A_MUX CONTROL
-------------------------------------------------------------------------
 MUXA_STATE_DECODE:process(count)
 begin
 next_muxa_state <= s20;
 case count is
  when 18 | 19 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 => next_muxa_state <= s21;
  when 20 => next_muxa_state <= s22;  -- ************
  when 23 => next_muxa_state <= s23;  -- ************
  when OTHERS => next_muxa_state <= s20;
 end case;
 end process MUXA_STATE_DECODE;

-------------------------------------------------------------------------
 MUXA_OUTPUT_DECODE:process(present_muxa_state)
 begin
  case present_muxa_state is
   when s20 => mux_sela <= "00";
   when s21 => mux_sela <= "01";
   when s22 => mux_sela <= "10";
   when s23 => mux_sela <= "11";
   when others => mux_sela <= "00"; -- Select A
  end case;
 end process MUXA_OUTPUT_DECODE;
```

```
-------------------------------------------------------------------------
-- B_MUX CONTROL
-------------------------------------------------------------------------
 MUXB_STATE_DECODE:process(count)
 begin
 next_muxb_state <= s20;
 case count is
  when 18 | 19 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 => next_muxb_state <= s21;
-- ***
  when 20 => next_muxb_state <= s22;  -- ***********
  when 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 => next_muxb_state <= s23;  -- ***
  when OTHERS => next_muxb_state <= s20;
 end case;
 end process MUXB_STATE_DECODE;

-------------------------------------------------------------------------
 MUXB_OUTPUT_DECODE:process(present_muxb_state)
 begin
  case present_muxb_state is
  when s20 => mux_selb <= "00";
  when s21 => mux_selb <= "01";
  when s22 => mux_selb <= "10";
  when s23 => mux_selb <= "11";
  when others => mux_selb <= "00"; -- Select A
  end case;
 end process MUXB_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- OUT_EN CONTROL
-------------------------------------------------------------------------
 MUXO_STATE_DECODE:process(count)
 begin
 next_muxo_state <= s10;
 case count is
  when 19 | 25 | 26 | 29 | 30 | 33 | 34 | 37 | 38 => next_muxo_state <= s11;  -- ***********
  when 23 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 => next_muxo_state <= s12;  -- ***********
  when OTHERS => next_muxo_state <= s10;
 end case;
 end process MUXO_STATE_DECODE;

-------------------------------------------------------------------------
 MUXO_OUTPUT_DECODE:process(present_muxo_state)
 begin
  case present_muxo_state is
  when s10 => mux_selo <= "00";
  when s11 => mux_selo <= "01";   -- Select Im | Yn |
  when s12 => mux_selo <= "10";   -- Select k
  when others => mux_selo <= "00"; -- Select Re | Yn |
```

```
 end case;
 end process MUXO_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- DLT TRIG
-------------------------------------------------------------------------
 DT_STATE_DECODE:process(count)
 begin
 next_trig2_state <= s40;
 case count is
  when 27 => next_trig2_state <= s41;  -- ***************************
  when 29 => next_trig2_state <= s42;  -- ***************************
  when 31 => next_trig2_state <= s43;  -- ***************************
  when 33 => next_trig2_state <= s44;  -- ***************************
  when 35 => next_trig2_state <= s45;  -- ***************************
  when 37 => next_trig2_state <= s46;  -- ***************************
  when 39 => next_trig2_state <= s47;  -- ***************************
  when 41 => next_trig2_state <= s48;  -- ***************************
  when OTHERS => next_trig2_state <= s40;
 end case;
 end process DT_STATE_DECODE;
-------------------------------------------------------------------------
 TRIG2_OUTPUT_DECODE:process(present_trig2_state)
 begin
  case present_trig2_state is
  when s41 => trig2 <= "00000001";
  when s42 => trig2 <= "00000010";
  when s43 => trig2 <= "00000100";
  when s44 => trig2 <= "00001000";
  when s45 => trig2 <= "00010000";
  when s46 => trig2 <= "00100000";
  when s47 => trig2 <= "01000000";
  when s48 => trig2 <= "10000000";
  when others => trig2 <= "00000000"; -- Trig nothing
  end case;
 end process TRIG2_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- ROW TRIG
-------------------------------------------------------------------------
 ROW_STATE_DECODE:process(count2)
 begin
 next_trig1_state <= s30;
 case count2 is
  when 47 => next_trig1_state <= s31;  -- ***************************
  when 48 => next_trig1_state <= s32;  -- ***************************
  when 49 => next_trig1_state <= s33;  -- ***************************
  when 50 => next_trig1_state <= s34;  -- ***************************
```

```
  when OTHERS => next_trig1_state <= s30;
 end case;
 end process ROW_STATE_DECODE;
-------------------------------------------------------------------------------
 TRIG1_OUTPUT_DECODE:process(present_trig1_state)
 begin
  case present_trig1_state is
   when s31 => trig1 <= "0001";
   when s32 => trig1 <= "0010";
   when s33 => trig1 <= "0100";
   when s34 => trig1 <= "1000";
   when others => trig1 <= "0000"; -- Trig nothing
  end case;
 end process TRIG1_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- SUB REG TRIG
-------------------------------------------------------------------------------
 WT_STATE_DECODE:process(count2)
 begin
 next_trig3_state <= s10;
 case count2 is
   when 41 | 44 | 47 | 50 => next_trig3_state <= s11;  -- ********************
   when 42 | 45 | 48 | 51 => next_trig3_state <= s12;  -- ******************
   when OTHERS => next_trig3_state <= s10;
 end case;
 end process WT_STATE_DECODE;
-------------------------------------------------------------------------------
 TRIG3_OUTPUT_DECODE:process(present_trig3_state)
 begin
  case present_trig3_state is
   when s11 => trig3 <= "01";
   when s12 => trig3 <= "10";
   when others => trig3 <= "00"; -- Trig nothing
  end case;
 end process TRIG3_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- OUT TRIG
-------------------------------------------------------------------------------
 OT_STATE_DECODE:process(count2)
 begin
 next_trig4_state <= s20;
 case count2 is
   when 10 => next_trig4_state <= s21;  -- ***************************
   when 18 => next_trig4_state <= s22;  -- ***************************
   when 22 | 25 => next_trig4_state <= s23;  -- ************************
   when OTHERS => next_trig4_state <= s20;
```

```
 end case;
 end process OT_STATE_DECODE;
-------------------------------------------------------------------------
 TRIG4_OUTPUT_DECODE:process(present_trig4_state)
 begin
  case present_trig4_state is
   when s21 => trig4 <= "001";
   when s22 => trig4 <= "010";
   when s23 => trig4 <= "100";
   when others => trig4 <= "000"; -- Trig nothing
  end case;
 end process TRIG4_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- INPUT CHANNEL REG TRIG
-------------------------------------------------------------------------
 INTRIG_STATE_DECODE:process(count)
 begin
 next_trig5_state <= s30;
 case count is
   when 1|2|3|4 => next_trig5_state <= s31;  -- ************************
   when 5|6|7|8 => next_trig5_state <= s32;  -- ************************
   when 9|10|11|12 => next_trig5_state <= s33;  -- **********************
   when 13|14|15|16 => next_trig5_state <= s34;  -- **********************
   when OTHERS => next_trig5_state <= s30;
 end case;
 end process INTRIG_STATE_DECODE;
-------------------------------------------------------------------------
 TRIG5_OUTPUT_DECODE:process(present_trig5_state)
 begin
  case present_trig5_state is
   when s31 => trig5 <= "0001";
   when s32 => trig5 <= "0010";
   when s33 => trig5 <= "0100";
   when s34 => trig5 <= "1000";
   when others => trig5 <= "0000"; -- Trig nothing
  end case;
 end process TRIG5_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- COMP CH REG TRIG
-------------------------------------------------------------------------
 COMP_STATE_DECODE:process(count2)
 begin
 next_trig6_state <= s00;
 case count2 is
   when 2|6|10|14 => next_trig6_state <= s01;  -- **************
   when OTHERS => next_trig6_state <= s00;
```

```
 end case;
 end process COMP_STATE_DECODE;
-------------------------------------------------------------------------
 TRIG6_OUTPUT_DECODE:process(present_trig6_state)
 begin
  case present_trig6_state is
   when s01 => trig6 <= '1';
   when others => trig6 <= '0'; -- Trig nothing
  end case;
 end process TRIG6_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- ACC EN
-------------------------------------------------------------------------
 ACC_STATE_DECODE:process(count2)
 begin
 next_en2_state <= s10;              -- Default: "11" = reset all
 case count2 is
  when 2|3|4|5|6|7|8|9|19|20|21|24|27|28|31|32|35|36|39|40 =>
       next_en2_state <= s11;  -- ***** "10" = ACC A
  when 10|11|12|13|14|15|16|17|25|26|29|30|33|34|37|38 =>
     next_en2_state <= s12;  -- ***** "01" = ACC B
  when OTHERS => next_en2_state <= s10;
 end case;
 end process ACC_STATE_DECODE;
-------------------------------------------------------------------------
 ACC_OUTPUT_DECODE:process(present_en2_state)
 begin
  case present_en2_state is
   when s10 => en2 <= "11";
   when s11 => en2 <= "10";
   when s12 => en2 <= "01";
   when others => en2 <= "11"; -- Reset all
  end case;
 end process ACC_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- AS CONT
-------------------------------------------------------------------------
 AS_STATE_DECODE:process(count2)
 begin
 next_en3_state <= s00;              --
 case count2 is
  when 3|5|7|9|28|32|36|40 => next_en3_state <= s01; -- '1' = SUBSTRUCT MODE
  when OTHERS => next_en3_state <= s00;
 end case;
 end process AS_STATE_DECODE;
-------------------------------------------------------------------------
```

```
 AS_OUTPUT_DECODE:process(present_en3_state)
 begin
  case present_en3_state is
   when s01 => en3 <= '0';
   when others => en3 <= '1'; -- Trig nothing
  end case;
 end process AS_OUTPUT_DECODE;

---------------------------------------------------------------------------
-- STATE REGISTER PROCESS
---------------------------------------------------------------------------
 MCLK2_STATE_REGISTER:process(mclk, start) --  Triggered by MCLK2
 begin
  if (start = '0') then
       present_trig1_state <= s30;
       present_trig3_state <= s10;
       present_trig4_state <= s20;
       present_trig6_state <= s00;
       present_en2_state <= s10;
       present_en3_state <= s00;
  elsif ( MCLK'EVENT AND MCLK = '0' AND MCLK'LAST_VALUE = '1') THEN
       present_trig1_state <= next_trig1_state;
       present_trig3_state <= next_trig3_state;
       present_trig4_state <= next_trig4_state;
       present_trig6_state <= next_trig6_state;
       present_en2_state <= next_en2_state;
       present_en3_state <= next_en3_state;
  end if;
 end process MCLK2_STATE_REGISTER;
---------------------------------------------------------------------------
 MCLK_STATE_REGISTER:process(mclk, start2) --  Triggered by MCLK
 begin
  if (start2 = '0') then
       present_mux1_state <= s0;
       present_mux2_state <= s20;
       present_mux3_state <= s0;
       present_mux4_state <= s20;
       present_mux5_state <= s00;
       present_muxa_state <= s20;
       present_muxb_state <= s20;
       present_muxo_state <= s10;
       present_trig2_state <= s40;
       present_trig5_state <= s30;
       present_we1_state <= s10;
  elsif ( MCLK'EVENT AND MCLK = '1' AND MCLK'LAST_VALUE = '0') THEN
       present_mux1_state <= next_mux1_state;
       present_mux2_state <= next_mux2_state;
       present_mux3_state <= next_mux3_state;
```

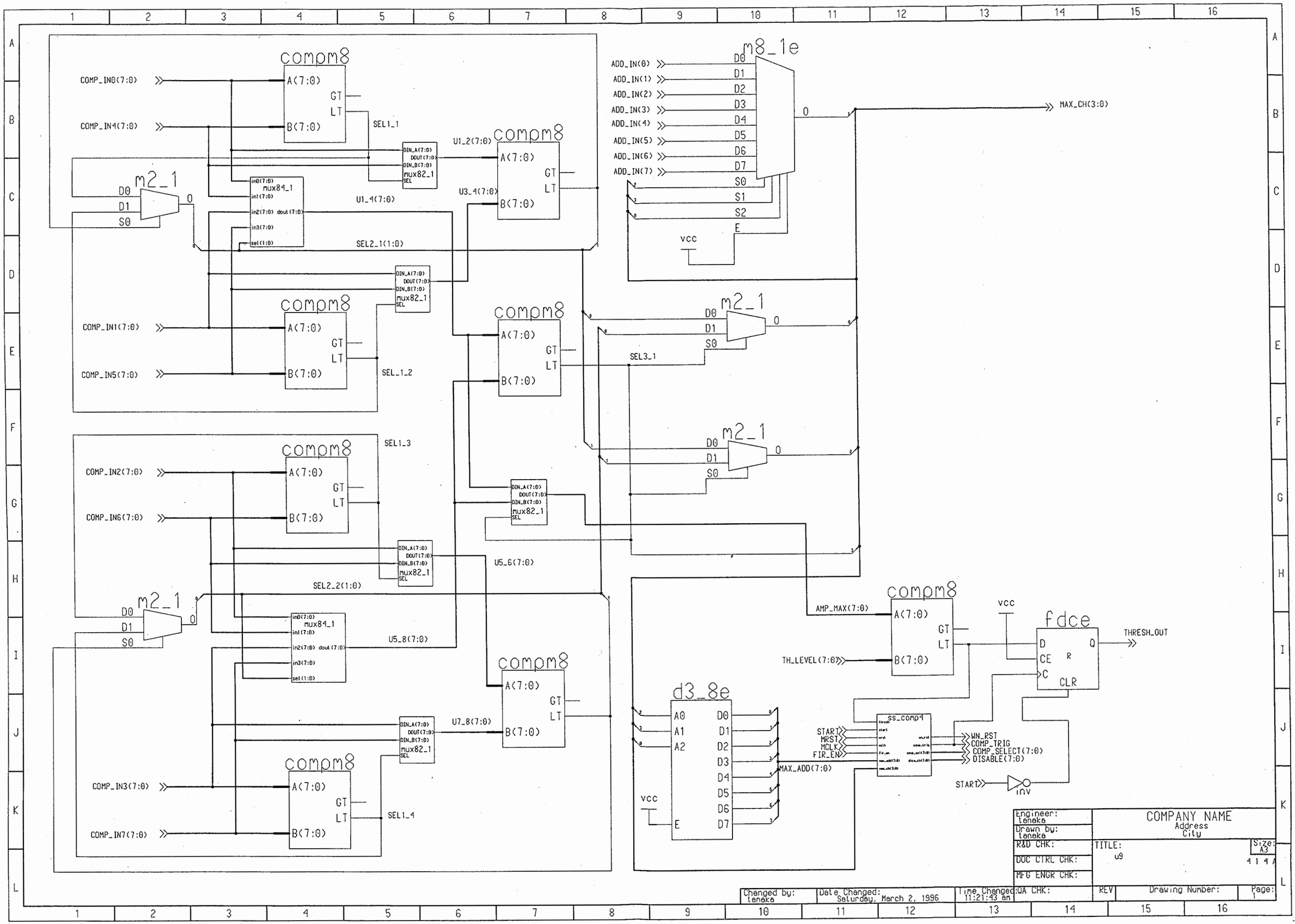
```
        present_mux4_state <= next_mux4_state;
        present_mux5_state <= next_mux5_state;
        present_muxa_state <= next_muxa_state;
        present_muxb_state <= next_muxb_state;
        present_muxo_state <= next_muxo_state;
        present_trig2_state <= next_trig2_state;
        present_trig5_state <= next_trig5_state;
        present_we1_state <= next_we1_state;
    end if;
  end process MCLK_STATE_REGISTER;

-------------------------------------------------------------------------
-- TRIGGERED BY MCLK, MCLK2 PROCESS = FINAL OUTPUT
-------------------------------------------------------------------------

  MCLK2_OUTPUT_REGISTER:process(MCLK, start) -- Triggered by MCLK2
  begin
  if ( start = '0' ) then
    ROW_MUX <= ( OTHERS => '0');
    WNR_MUX <= ( OTHERS => '0');
    DLT_MUX <= ( OTHERS => '0');
    WNI_MUX <= ( OTHERS => '0');
    WNRI_MUX <= '0';
    A_MUX <= ( OTHERS => '0');
    B_MUX <= ( OTHERS => '0');
    OUT_MUX <= "00";
    DLT_TRIG <= ( OTHERS => '0');
    CHREG_TRIG <= ( OTHERS => '0');
    WN_RAM_WE <= ( OTHERS => '0');
  elsif ( MCLK'EVENT AND MCLK = '0' AND MCLK'LAST_VALUE = '1' ) then
    ROW_MUX <= mux_sel1;
    WNR_MUX <= mux_sel2;
    DLT_MUX <= mux_sel3;
    WNI_MUX <= mux_sel4;
    WNRI_MUX <= mux_sel5;
    A_MUX <= mux_sela;
    B_MUX <= mux_selb;
    OUT_MUX <= mux_selo;
    DLT_TRIG <= trig2;
    CHREG_TRIG <= trig5;
    WN_RAM_WE <= we1;
  end if;
  end process MCLK2_OUTPUT_REGISTER;

-------------------------------------------------------------------------

  MCLK_OUTPUT_REGISTER:process(MCLK, START2) -- Triggered by MCLK
  begin
```

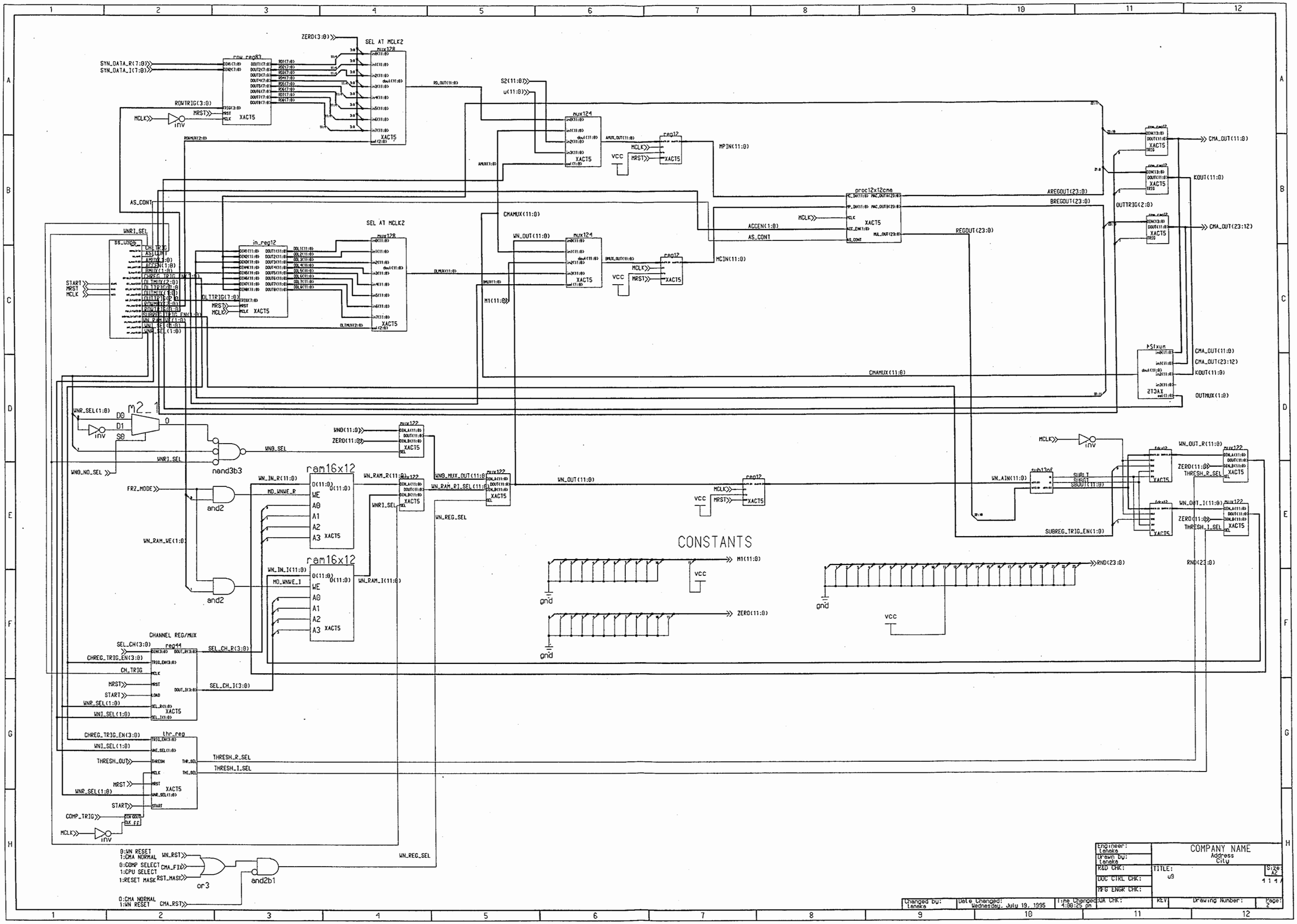
```
  if ( START2 = '0' ) then
    OUT_TRIG <= ( OTHERS => '0');
    SUBREG_TRIG <= ( OTHERS => '0');
    ROW_TRIG <= ( OTHERS => '0');
    COMP_TRIG <= '0';
    ACC_EN <= ( OTHERS => '1');
    AS_CONT <= '0';
  elsif ( MCLK'EVENT AND MCLK = '1' AND MCLK'LAST_VALUE = '0' ) then
    OUT_TRIG <= trig4;
    SUBREG_TRIG <= trig3;
    ROW_TRIG <= trig1;
    COMP_TRIG <= trig6;
    ACC_EN <= en2;
    AS_CONT <= en3;
  end if;
  end process MCLK_OUTPUT_REGISTER;

-------------------------------------------------------------------------
-- End of rtl.
-------------------------------------------------------------------------
end rtl;
```

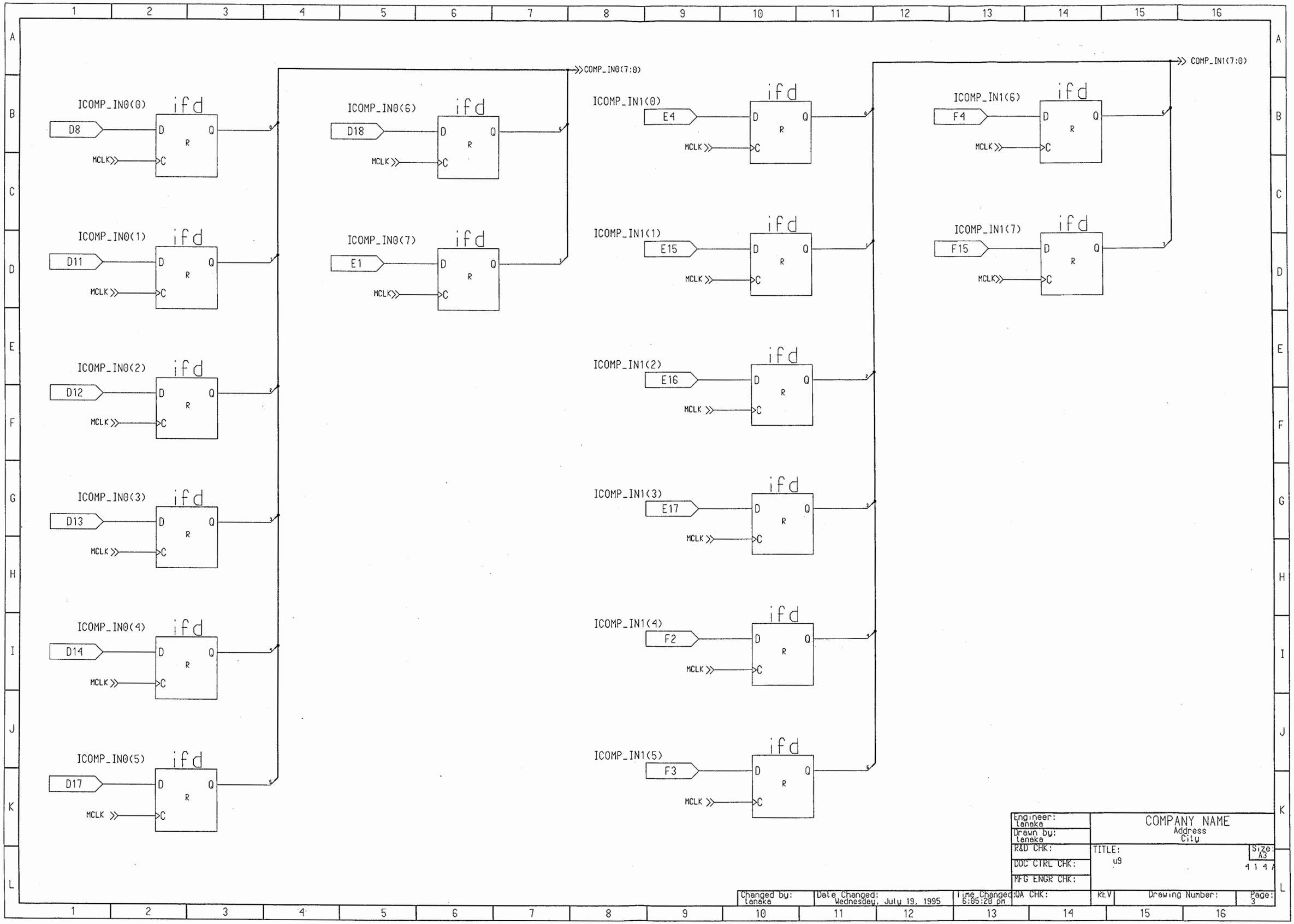
第 2 項　sub13of VHDL Listing

```
library mgc_portable;
use mgc_portable.qsim_logic.all;

-- sub9of Entity Description
entity sub9of3 is
  port(
    A: in qsim_state_vector(7 downto 0);
    B: in qsim_state_vector(8 downto 0);
    D: out qsim_state_vector(7 downto 0);
    GT,LT: out qsim_state
  );
end sub9of3;

-- sub9of3 Architecture Description
architecture rtl of sub9of3 is
    signal pre_D : qsim_state_vector(8 downto 0);
    signal pre_OV : qsim_state;
begin
  ARITHMETIC_Process: process(A,B)
    variable fct_out : qsim_state_vector(8 downto 0);
    variable a_ext,b_ext : qsim_state_vector(8 downto 0);
    variable pre_B : qsim_state_vector(8 downto 0);
    variable carry_ext : qsim_state_vector(1 downto 0);
    variable msb : integer;
```

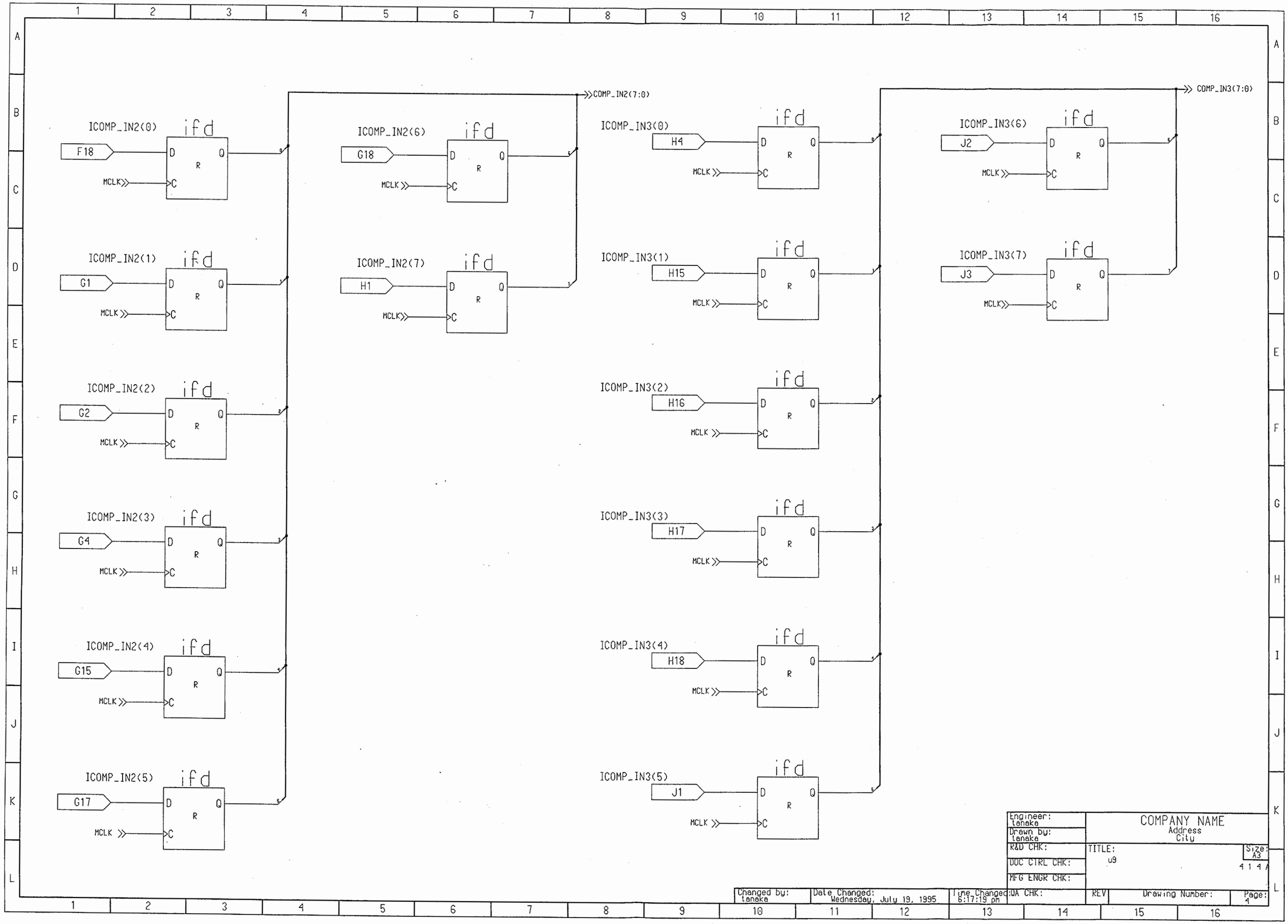
```
  begin
    -- pre b process rounding LSB
    carry_ext := (OTHERS => '0');
    carry_ext(0) := '1';
    pre_B := B + carry_ext;

    -- zero extend inputs to include carry bit
    a_ext := '0' & A;
    b_ext := '0' & not pre_b(8 downto 1);

    -- SUB
    fct_out := a_ext + b_ext + carry_ext;

    -- Assign to signal for use outside process
    pre_D <= fct_out;

    -- Calculate overflow bit
    if (a_ext(7) = b_ext(7) and fct_out(7) = not a_ext(7)) then
      pre_OV <= '1';
    else
      pre_OV <= '0';
    end if;
  end process ARITHMETIC_Process;

  -- Assign the outputs
  D <= pre_D(7 downto 0);

  -- Assign flags
  GT <= pre_OV and pre_D(7);
  LT <= pre_OV and not pre_D(7);
end rtl;
```

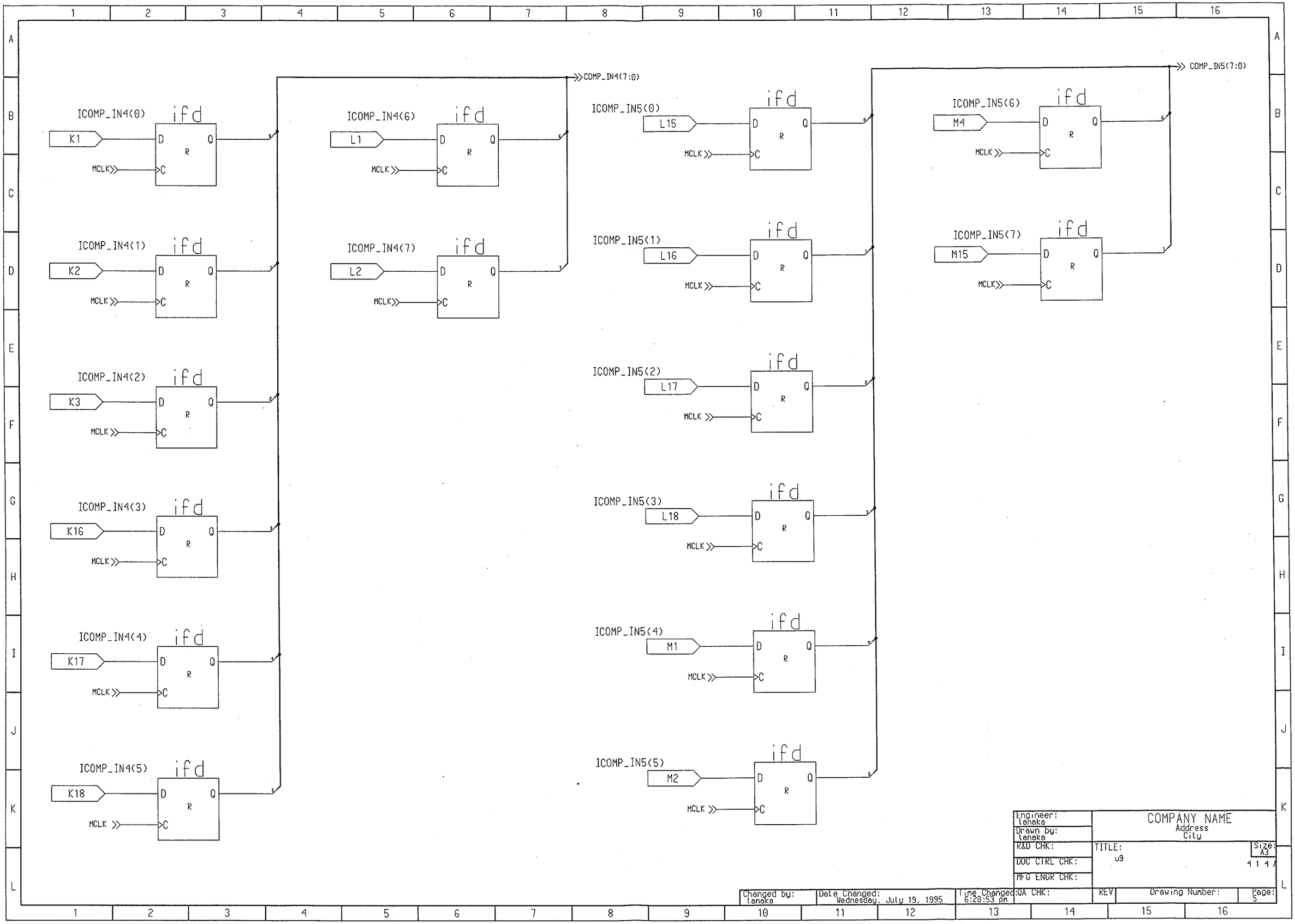

compm8
COMP_IN0(7:0)
COMP_IN4(7:0)
A(7:0)
B(7:0)
GT
LT
SEL1_1
m2_1
D0
D1
S0
O
mux84_1
mux82_1
DIN_A(7:0)
DIN_B(7:0)
DOUT(7:0)
SEL
U1_2(7:0)
U1_4(7:0)
U3_4(7:0)
SEL2_1(1:0)
COMP_IN1(7:0)
COMP_IN5(7:0)
SEL_1_2
SEL1_3
COMP_IN2(7:0)
COMP_IN6(7:0)
U5_6(7:0)
SEL2_2(1:0)
U5_8(7:0)
COMP_IN3(7:0)
COMP_IN7(7:0)
SEL1_4
U7_8(7:0)
SEL3_1
m8_1e
ADD_IN(0)
ADD_IN(1)
ADD_IN(2)
ADD_IN(3)
ADD_IN(4)
ADD_IN(5)
ADD_IN(6)
ADD_IN(7)
S1
S2
E
VCC
MAX_CH(3:0)
d3_8e
A0
A1
A2
MAX_ADD(7:0)
AMP_MAX(7:0)
TH_LEVEL(7:0)
fdce
D
CE
C
R
CLR
Q
THRESH_OUT
ss_comp1
START
MRST
MCLK
FIR_EN
WN_RST
COMP_TRIG
COMP_SELECT(7:0)
DISABLE(7:0)
inv
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
QA CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
REV
Drawing Number:
Page: 1
Changed by: tanaka
Date Changed: Saturday, March 2, 1996
Time Changed: 11:21:43 am

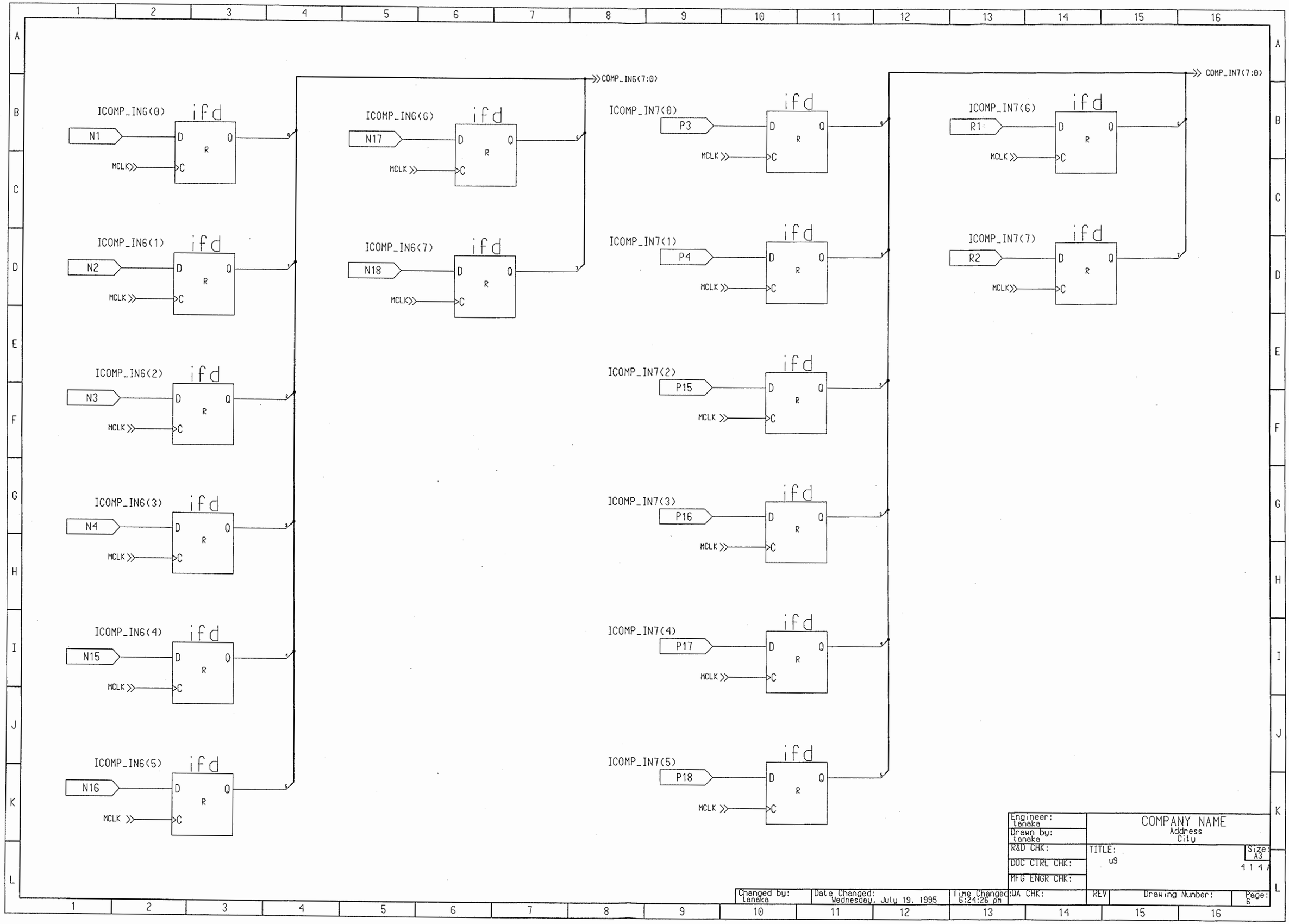

SYN_DATA_R(7:0)
SYN_DATA_I(7:0)
row_reg83
ZERO(3:0)
SEL AT MCLK2
mux128
RO_OUT(11:0)
ROWTRIG(3:0)
MCLK
MRST
inv
XACT5
ROWMUX(2:0)
S2(11:0)
u(11:0)
mux124
AMUX_OUT(11:0)
reg12
VCC
MPIN(11:0)
AMUX(1:0)
AS_CONT
CHAMUX(11:0)
WNRI_SEL
SS_USER
CH_TRIG
AMUX(3:0)
ACCEN(1:0)
FMUX(1:0)
CHREG_TRIG_EN(3:0)
OUTMUX(2:0)
OUTTRIG(2:0)
ROWMUX(2:0)
SUPPRESS_TRIG_EN(1:0)
WN_RAM_WE(1:0)
WNI_SEL(1:0)
WNR_SEL(1:0)
START
in_reg12
mux128
SEL AT MCLK2
OLTRIG(7:0)
OLTMUX(2:0)
WN_OUT(11:0)
BMUX_OUT(11:0)
MCIN(11:0)
M1(11:0)
proc12x12cma
MP_IN(11:0)
MAC_OUTA(23:0)
MAC_OUTB(23:0)
ACC_EN(1:0)
MAC_OUT(23:0)
REGOUT(23:0)
AREGOUT(23:0)
BREGOUT(23:0)
CMA_OUT(11:0)
KOUT(11:0)
CMA_OUT(23:12)
OUTTRIG(2:0)
mux154
OUTMUX(1:0)
WNR_SEL(1:0)
D0
D1
S0
m2_1
nand3b3
WN0_SEL
WN0_NO_SEL
WN0(11:0)
ZERO(11:0)
mux122
WN0_MUX_OUT(11:0)
WN_RAM_R1_SEL(11:0)
WNR1_SEL
WN_REG_SEL
FRZ_MODE
and2
WN_IN_R(11:0)
WN_RAM_R(11:0)
MD_WNWE_R
ram16x12
WN_RAM_WE(1:0)
WN_IN_I(11:0)
WN_RAM_I(11:0)
MD_WNWE_I
CHANNEL REG/MUX
SEL_CH(3:0)
reg41
CHREG_TRIG_EN(3:0)
CH_TRIG
SEL_CH_R(3:0)
SEL_CH_I(3:0)
thr_reg
THRESH_OUT
THRESH_R_SEL
THRESH_I_SEL
COMP_TRIG
MCLK
inv
0:WN RESET
1:CMA NORMAL
0:COMP SELECT
1:CPU SELECT
1:RESET MASK
0:CMA NORMAL
1:WN RESET
WN_RST
CMA_FIX
RST_MASK
CMA_RST
or3
and2b1
CONSTANTS
M1(11:0)
RND(23:0)
gnd
vcc
WN_AIN(11:0)
sub13af
SUBRT
SUBOUT(11:0)
SUBREG_TRIG_EN(1:0)
WN_OUT_R(11:0)
WN_OUT_I(11:0)
THRESH_R_SEL
THRESH_I_SEL
COMPANY NAME
Address
City
Engineer: Taneka
Drawn by: Taneka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
TITLE: u9
Size: A2
Changed by: Taneka
Date Changed: Wednesday, July 19, 1995
Time Changed: 4:00:25 pm
QA CHK:
REV
Drawing Number:
Page: 2

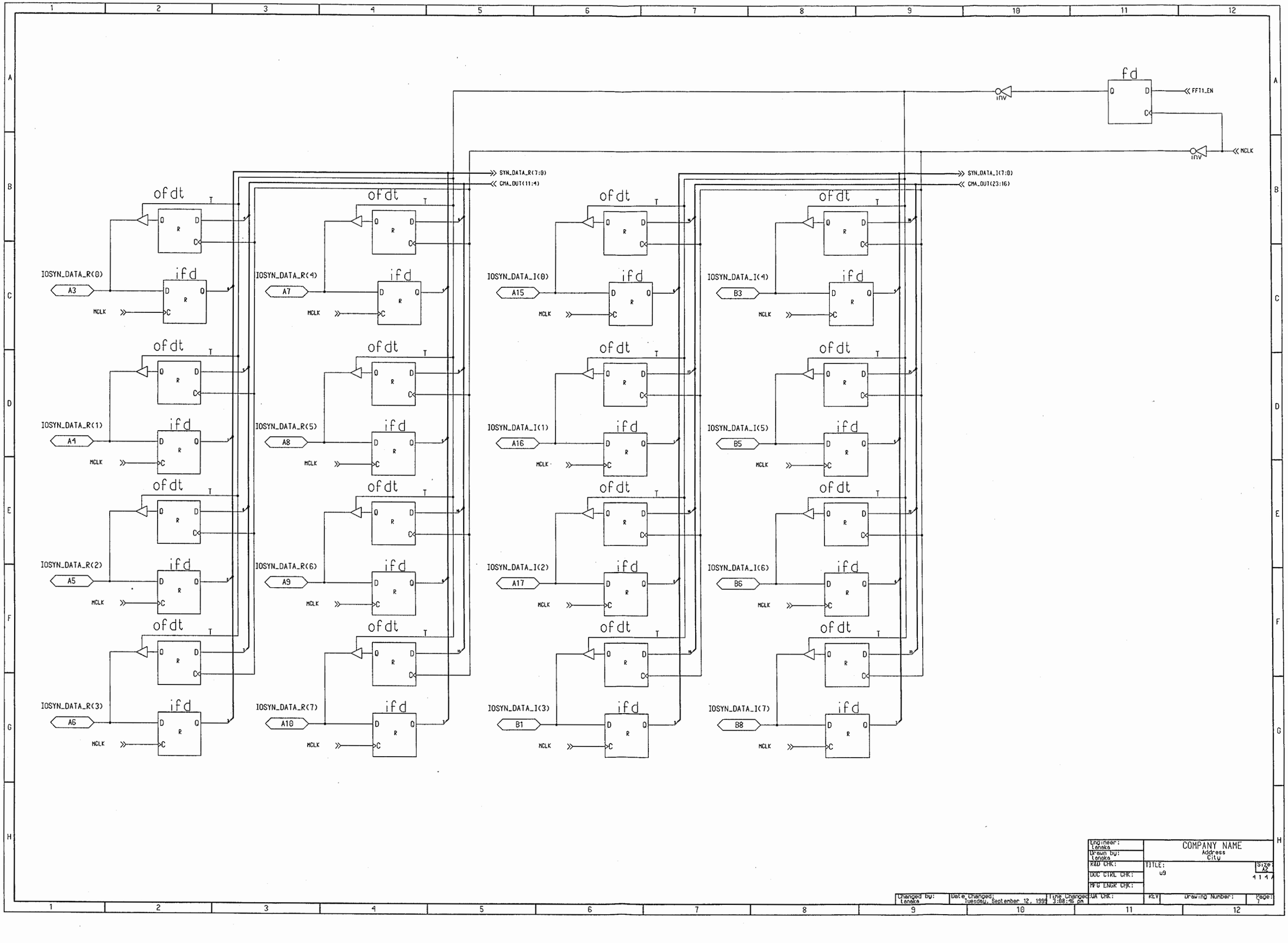


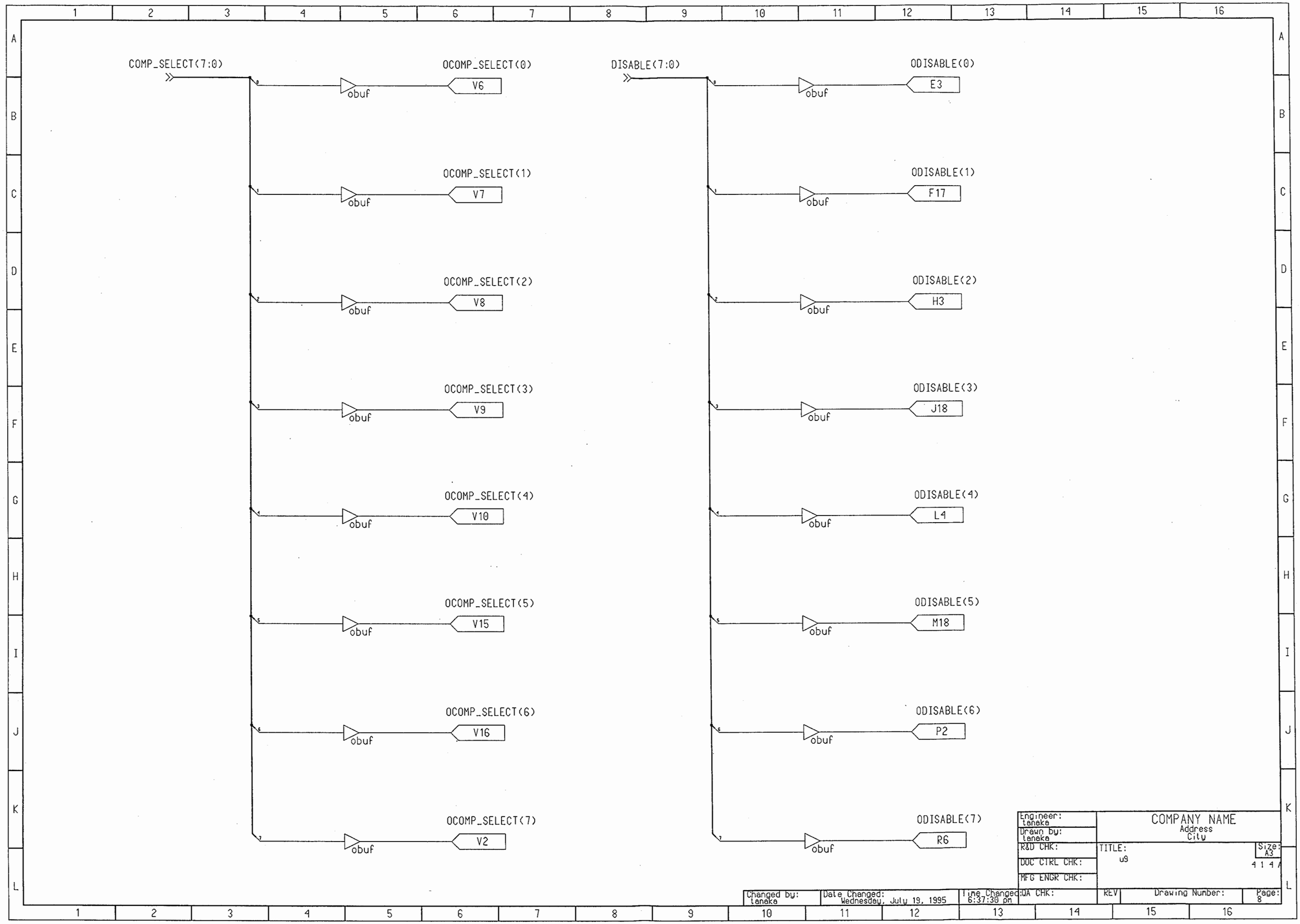
Engineer: tanaka	COMPANY NAME		
Drawn by: tanaka	Address		
R&D CHK:	City		
DOC CTRL CHK:	TITLE: u9	Size: A3	
MFG ENGR CHK:		4 1 4 A	
Changed by: tanaka	Date Changed: Wednesday, July 19, 1995	Time Changed: 6:05:20 pm	QA CHK:
REV	Drawing Number:	Page: 3	

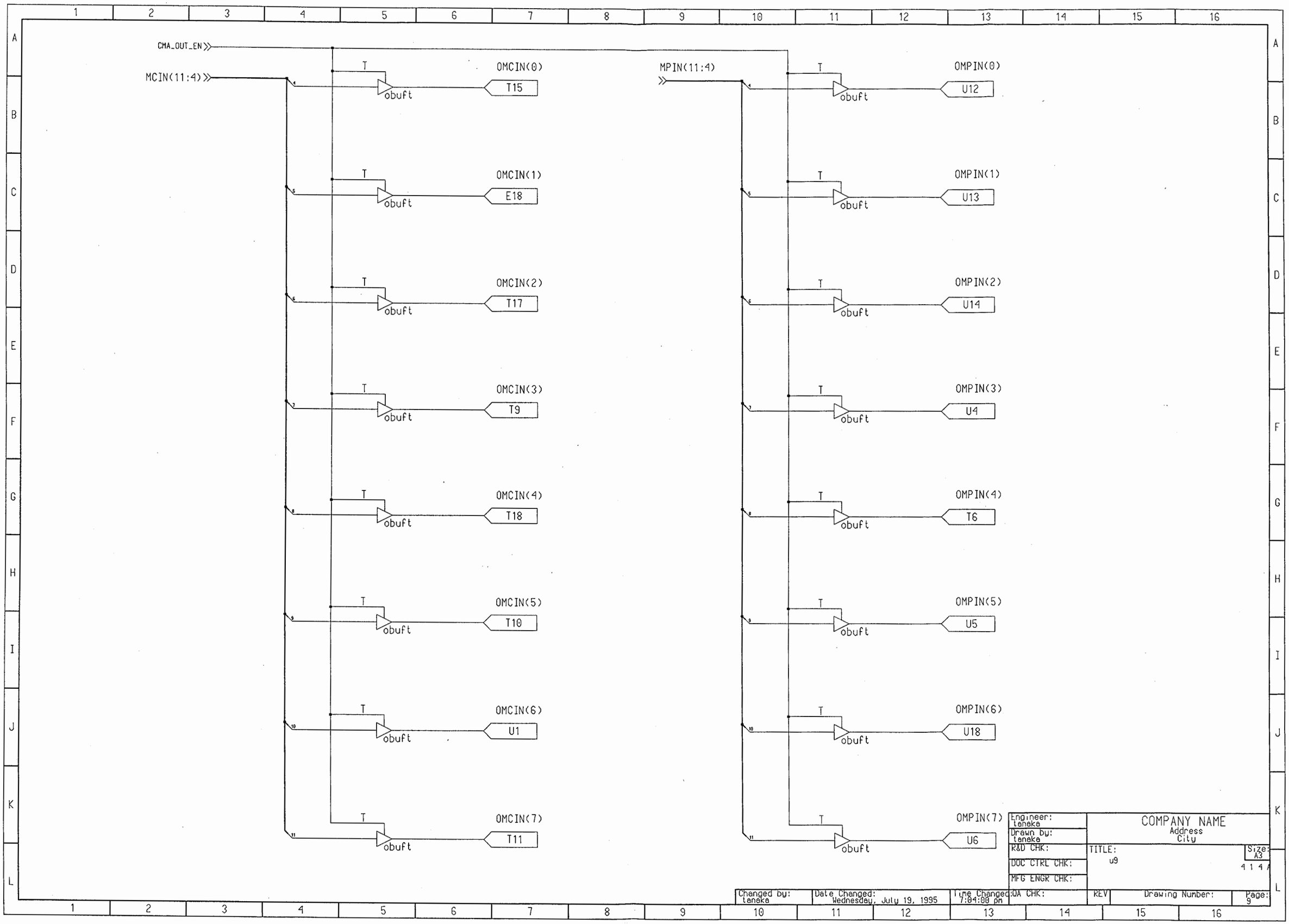

ICOMP_IN2(0)
F18
ICOMP_IN2(1)
G1
ICOMP_IN2(2)
G2
ICOMP_IN2(3)
G4
ICOMP_IN2(4)
G15
ICOMP_IN2(5)
G17
ICOMP_IN2(6)
G18
ICOMP_IN2(7)
H1
COMP_IN2(7:0)
ICOMP_IN3(0)
H4
ICOMP_IN3(1)
H15
ICOMP_IN3(2)
H16
ICOMP_IN3(3)
H17
ICOMP_IN3(4)
H18
ICOMP_IN3(5)
J1
ICOMP_IN3(6)
J2
ICOMP_IN3(7)
J3
COMP_IN3(7:0)
ifd
D
Q
R
C
MCLK
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
Changed by: tanaka
Date Changed: Wednesday, July 19, 1995
Time Changed: 6:17:19 pm
QA CHK:
REV
Drawing Number:
Page: 4

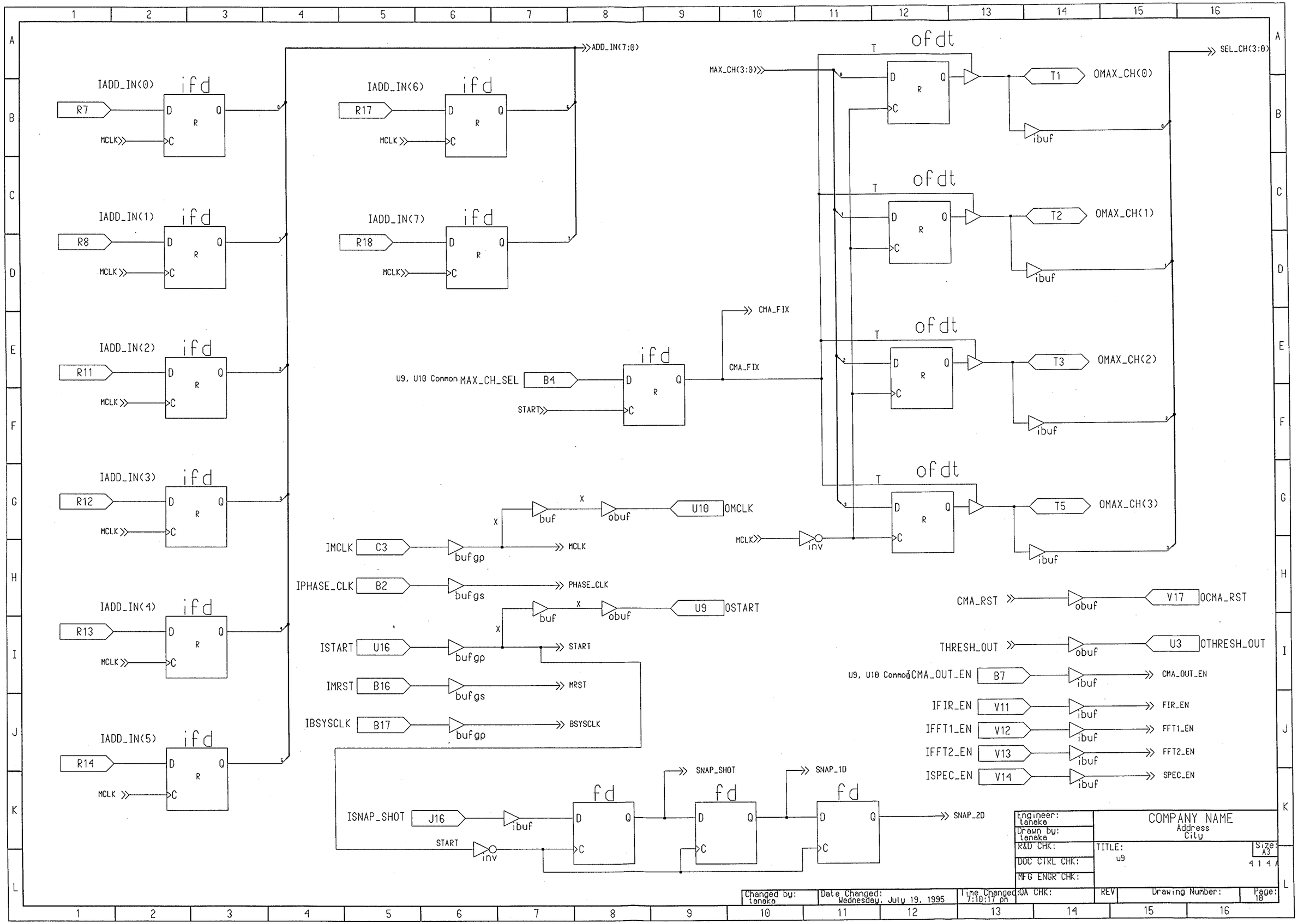

ICOMP_IN4(0)
K1
ICOMP_IN4(1)
K2
ICOMP_IN4(2)
K3
ICOMP_IN4(3)
K16
ICOMP_IN4(4)
K17
ICOMP_IN4(5)
K18
ICOMP_IN4(6)
L1
ICOMP_IN4(7)
L2
COMP_IN4(7:0)
ICOMP_IN5(0)
L15
ICOMP_IN5(1)
L16
ICOMP_IN5(2)
L17
ICOMP_IN5(3)
L18
ICOMP_IN5(4)
M1
ICOMP_IN5(5)
M2
ICOMP_IN5(6)
M4
ICOMP_IN5(7)
M15
COMP_IN5(7:0)
ifd
D
Q
R
C
MCLK
Changed by: teneke
Date Changed: Wednesday, July 19, 1995
Time Changed: 6:20:53 pm
Engineer: teneke
Drawn by: teneke
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
QA CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
REV
Drawing Number:
Page: 5

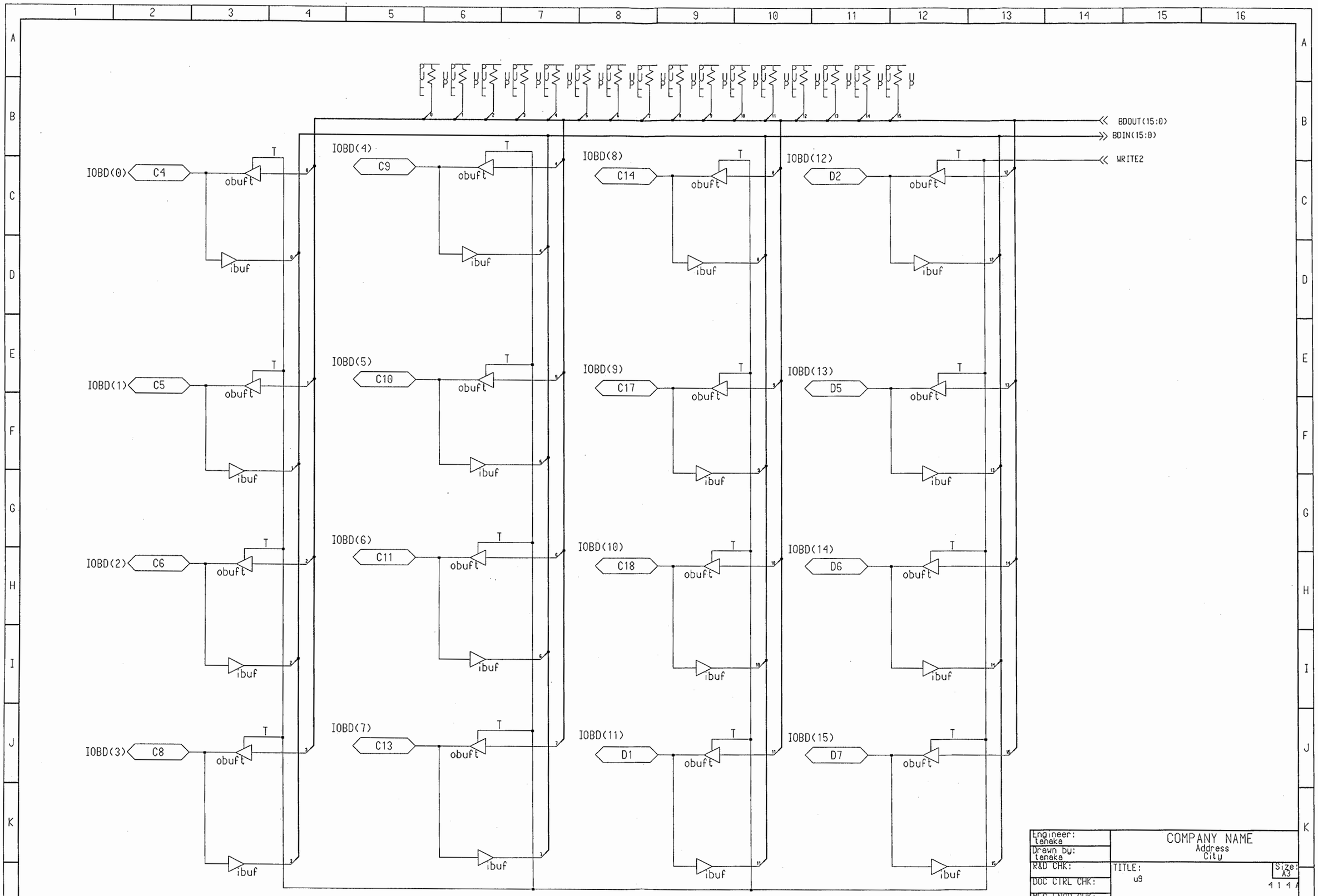

ICOMP_IN6(0)
N1
MCLK
ICOMP_IN6(1)
N2
MCLK
ICOMP_IN6(2)
N3
MCLK
ICOMP_IN6(3)
N4
MCLK
ICOMP_IN6(4)
N15
MCLK
ICOMP_IN6(5)
N16
MCLK
ICOMP_IN6(6)
N17
MCLK
ICOMP_IN6(7)
N18
MCLK
COMP_IN6(7:0)
ICOMP_IN7(0)
P3
MCLK
ICOMP_IN7(1)
P4
MCLK
ICOMP_IN7(2)
P15
MCLK
ICOMP_IN7(3)
P16
MCLK
ICOMP_IN7(4)
P17
MCLK
ICOMP_IN7(5)
P18
MCLK
ICOMP_IN7(6)
R1
MCLK
ICOMP_IN7(7)
R2
MCLK
COMP_IN7(7:0)
ifd
D
Q
R
C
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
Changed by: tanaka
Date Changed: Wednesday, July 19, 1995
Time Changed: 6:24:26 pm
QA CHK:
REV
Drawing Number:
Page: 6

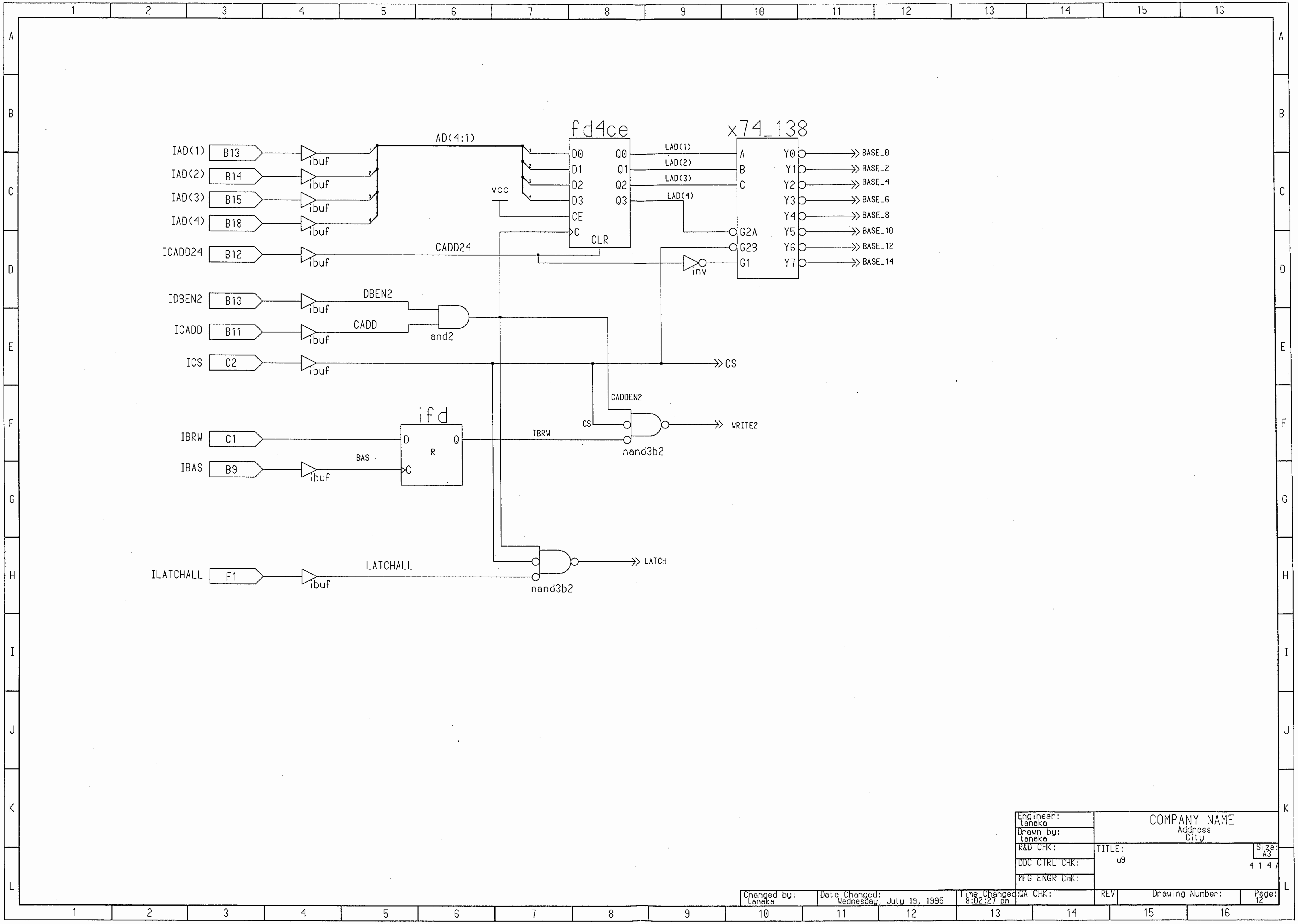

fd
inv
FFT1_EN
MCLK
SYN_DATA_R(7:0)
CMA_OUT(11:4)
SYN_DATA_I(7:0)
CMA_OUT(23:16)
ofdt
ifd
IOSYN_DATA_R(0)
A3
IOSYN_DATA_R(4)
A7
IOSYN_DATA_I(0)
A15
IOSYN_DATA_I(4)
B3
IOSYN_DATA_R(1)
A4
IOSYN_DATA_R(5)
A8
IOSYN_DATA_I(1)
A16
IOSYN_DATA_I(5)
B5
IOSYN_DATA_R(2)
A5
IOSYN_DATA_R(6)
A9
IOSYN_DATA_I(2)
A17
IOSYN_DATA_I(6)
B6
IOSYN_DATA_R(3)
A6
IOSYN_DATA_R(7)
A10
IOSYN_DATA_I(3)
B1
IOSYN_DATA_I(7)
B8
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A2
Changed by: tanaka
Date Changed: Tuesday, September 12, 1999
Time Changed: 3:08:46 pm
QA CHK:
REV
Drawing Number:
Page: 7


COMP_SELECT(7:0)
OCOMP_SELECT(0)
V6
OCOMP_SELECT(1)
V7
OCOMP_SELECT(2)
V8
OCOMP_SELECT(3)
V9
OCOMP_SELECT(4)
V10
OCOMP_SELECT(5)
V15
OCOMP_SELECT(6)
V16
OCOMP_SELECT(7)
V2
DISABLE(7:0)
ODISABLE(0)
E3
ODISABLE(1)
F17
ODISABLE(2)
H3
ODISABLE(3)
J18
ODISABLE(4)
L4
ODISABLE(5)
M18
ODISABLE(6)
P2
ODISABLE(7)
R6
obuf
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
Changed by: tanaka
Date Changed: Wednesday, July 19, 1995
Time Changed: 6:37:30 pm
QA CHK:
REV
Drawing Number:
Page: 8


CMA_OUT_EN
MCIN(11:4)
MPIN(11:4)
T
obuft
OMCIN(0)
T15
OMCIN(1)
E18
OMCIN(2)
T17
OMCIN(3)
T9
OMCIN(4)
T18
OMCIN(5)
T10
OMCIN(6)
U1
OMCIN(7)
T11
OMPIN(0)
U12
OMPIN(1)
U13
OMPIN(2)
U14
OMPIN(3)
U4
OMPIN(4)
T6
OMPIN(5)
U5
OMPIN(6)
U18
OMPIN(7)
U6
Engineer: teneke
Drawn by: teneke
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
Changed by: teneke
Date Changed: Wednesday, July 19, 1995
Time Changed: 7:04:00 pm
QA CHK:
REV
Drawing Number:
Page: 9

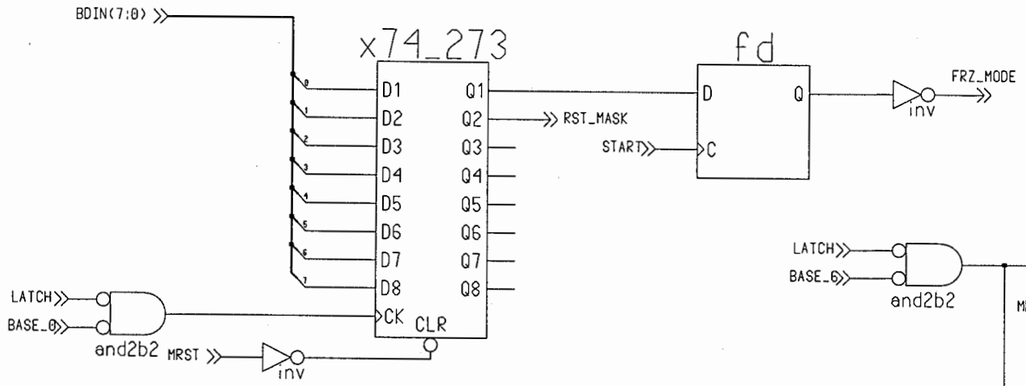

IADD_IN(0)
R7
MCLK
IADD_IN(1)
R8
IADD_IN(2)
R11
IADD_IN(3)
R12
IADD_IN(4)
R13
IADD_IN(5)
R14
IADD_IN(6)
R17
IADD_IN(7)
R18
ifd
D
Q
R
C
ADD_IN(7:0)
MAX_CH(3:0)
ofdt
T
T1
T2
T3
T5
OMAX_CH(0)
OMAX_CH(1)
OMAX_CH(2)
OMAX_CH(3)
ibuf
SEL_CH(3:0)
U9, U10 Common MAX_CH_SEL
B4
START
CMA_FIX
IMCLK
C3
bufgp
buf
obuf
U10
OMCLK
IPHASE_CLK
B2
bufgs
PHASE_CLK
U9
OSTART
ISTART
U16
MRST
B16
IBSYSCLK
B17
BSYSCLK
CMA_RST
V17
OCMA_RST
THRESH_OUT
U3
OTHRESH_OUT
CMA_OUT_EN
B7
IFIR_EN
V11
IFFT1_EN
V12
IFFT2_EN
V13
ISPEC_EN
V14
FIR_EN
FFT1_EN
FFT2_EN
SPEC_EN
ISNAP_SHOT
J16
SNAP_SHOT
SNAP_1D
SNAP_2D
fd
inv
Engineer: Tenaka
Drawn by: Tenaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
Changed by: tenaka
Date Changed: Wednesday, July 19, 1995
Time Changed: 7:10:17 pm
QA CHK:
REV
Drawing Number:
Page: 10


BDOUT(15:0)
BDIN(15:0)
WRITE2
IOBD(0)
C4
IOBD(4)
C9
IOBD(8)
C14
IOBD(12)
D2
IOBD(1)
C5
IOBD(5)
C10
IOBD(9)
C17
IOBD(13)
D5
IOBD(2)
C6
IOBD(6)
C11
IOBD(10)
C18
IOBD(14)
D6
IOBD(3)
C8
IOBD(7)
C13
IOBD(11)
D1
IOBD(15)
D7
T
obuft
ibuf
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
QA CHK:
COMPANY NAME
Address
City
TITLE: u9
Size: A3
4 1 4 A
Changed by: tanaka
Date Changed: Wednesday, July 19, 1995
Time Changed: 7:59:07 pm
REV
Drawing Number:
Page: 11

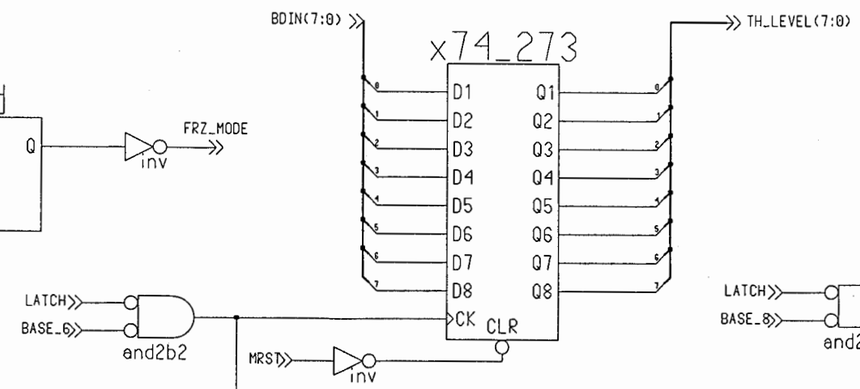

fd4ce
x74_138
AD(4:1)
IAD(1) B13 ibuf
IAD(2) B14 ibuf
IAD(3) B15 ibuf
IAD(4) B18 ibuf
ICADD24 B12 ibuf
CADD24
D0 Q0
D1 Q1
D2 Q2
D3 Q3
CE
C
CLR
VCC
LAD(1)
LAD(2)
LAD(3)
LAD(4)
A Y0
B Y1
C Y2
Y3
Y4
G2A Y5
G2B Y6
G1 Y7
inv
BASE_0
BASE_2
BASE_4
BASE_6
BASE_8
BASE_10
BASE_12
BASE_14
IDBEN2 B10 ibuf
DBEN2
ICADD B11 ibuf
CADD
and2
ICS C2 ibuf
CS
CADDEN2
CS
WRITE2
nand3b2
ifd
IBRW C1
D Q
R
C
TBRW
IBAS B9 ibuf
BAS
ILATCHALL F1 ibuf
LATCHALL
LATCH
nand3b2
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
QA CHK:
COMPANY NAME
Address
City
TITLE: u9
Size: A3
REV
Drawing Number:
Page: 12
Changed by: tanaka
Date Changed: Wednesday, July 19, 1995
Time Changed: 8:02:27 pm

WRITE REGISTER

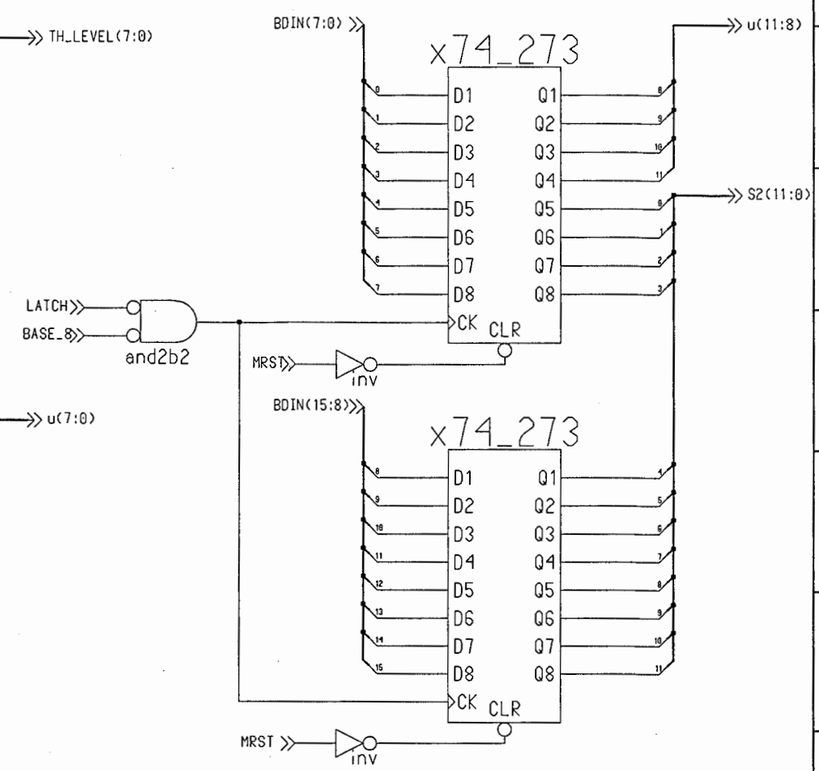
BASE_0



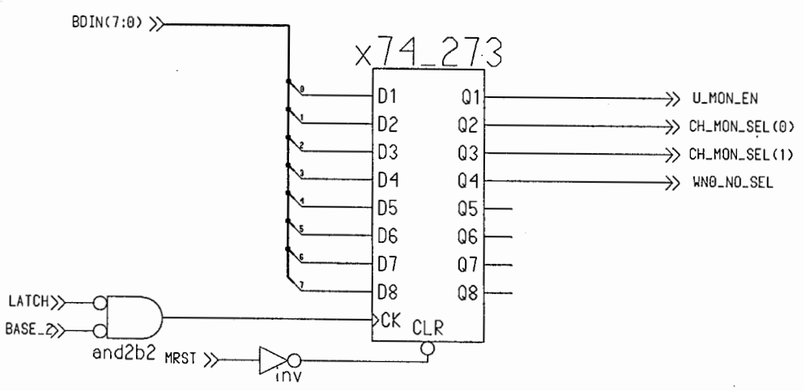
BASE_6



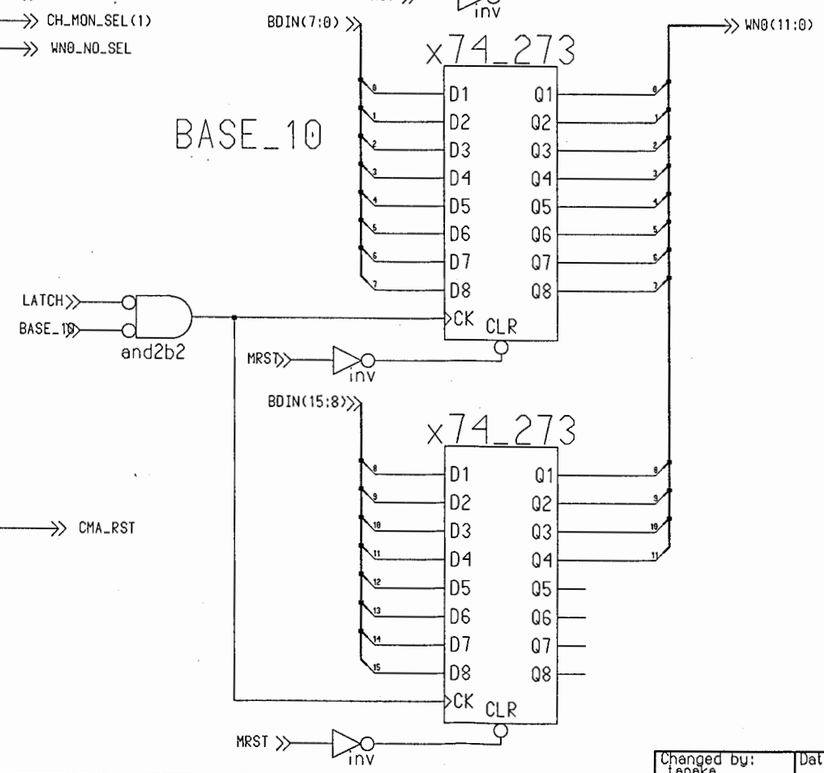
BASE_8



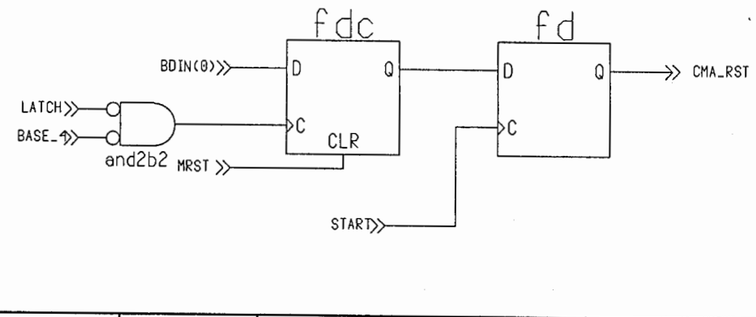
BASE_2



BASE_10



BASE_4

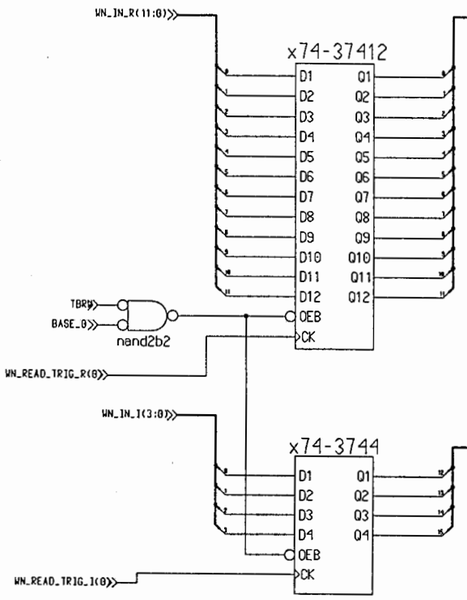


Engineer: taneka	COMPANY NAME
Drawn by: taneka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: u9
MFG ENGR CHK:	Size: A3
QA CHK:	REV / Drawing Number: / Page: 13

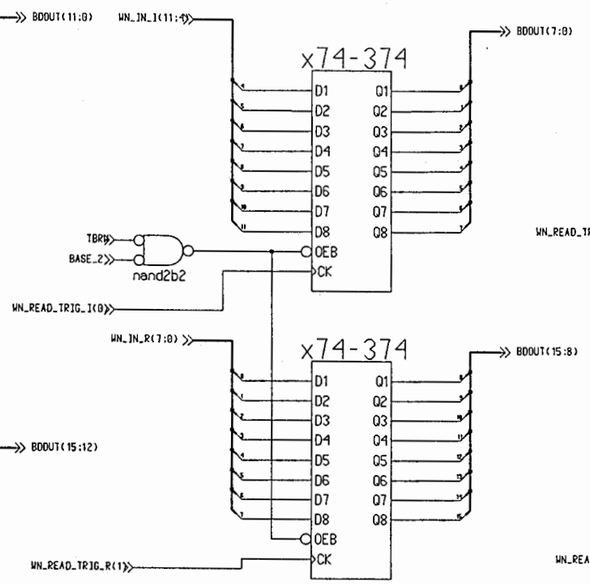
Changed by: taneka | Date Changed: Wednesday, July 19, 1995 | Time Changed: 8:25:19 pm

READ REGISTER

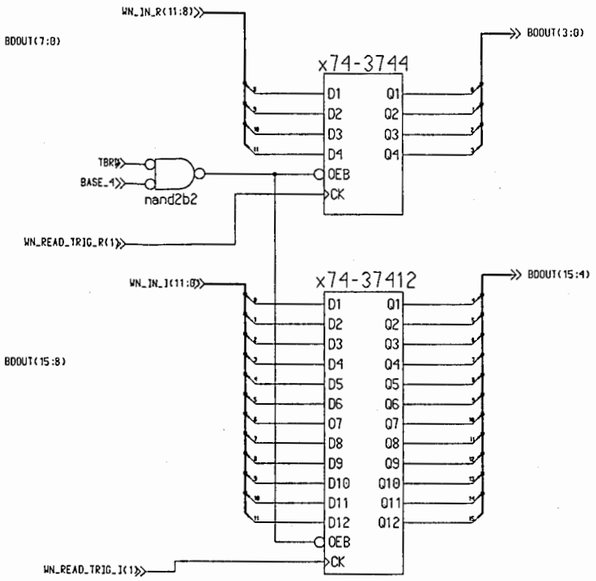
BASE +0 WN(1)R/I REGISTER



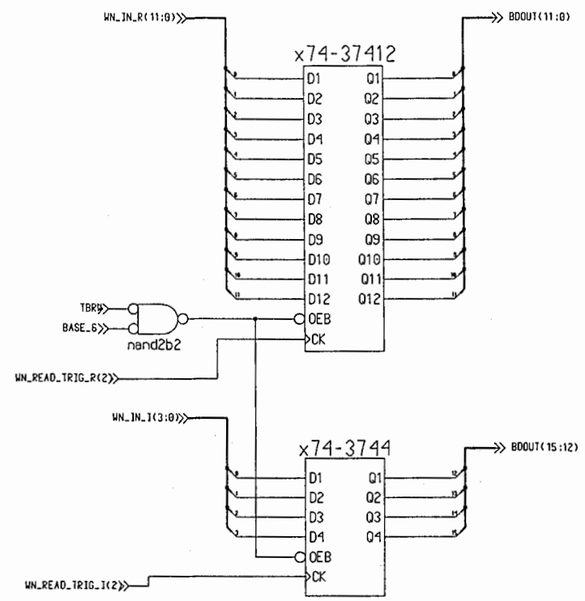
BASE +2 WN(1)I/WN(2)R REGISTER

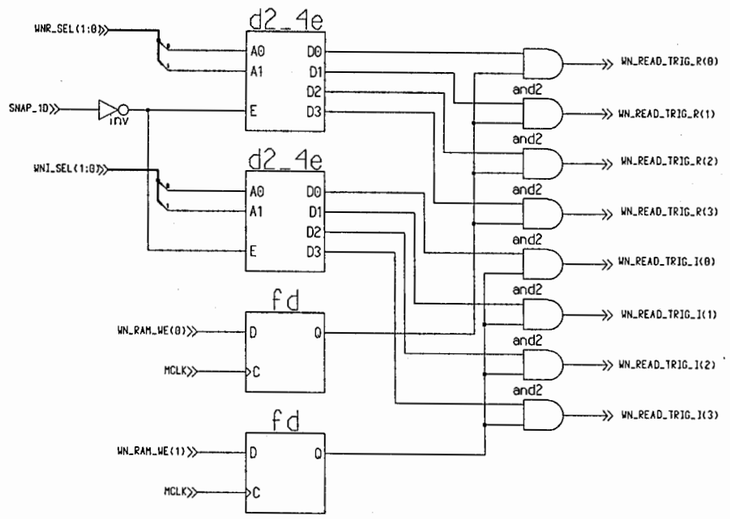


BASE +4 WN(2)R/I REGISTER

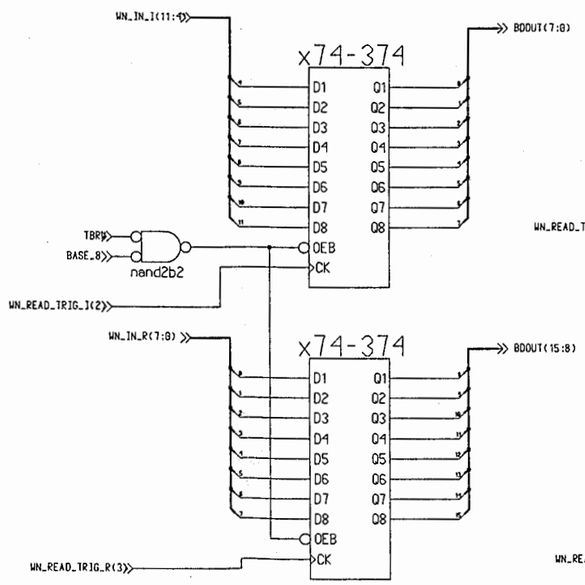


BASE +6 WN(3)R/I REGISTER

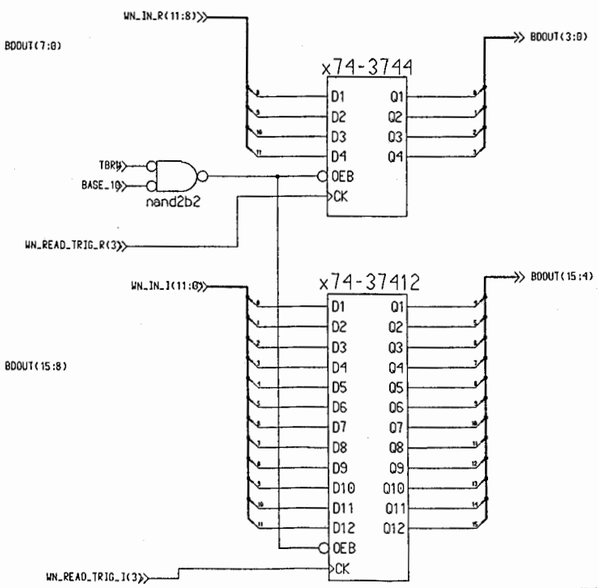




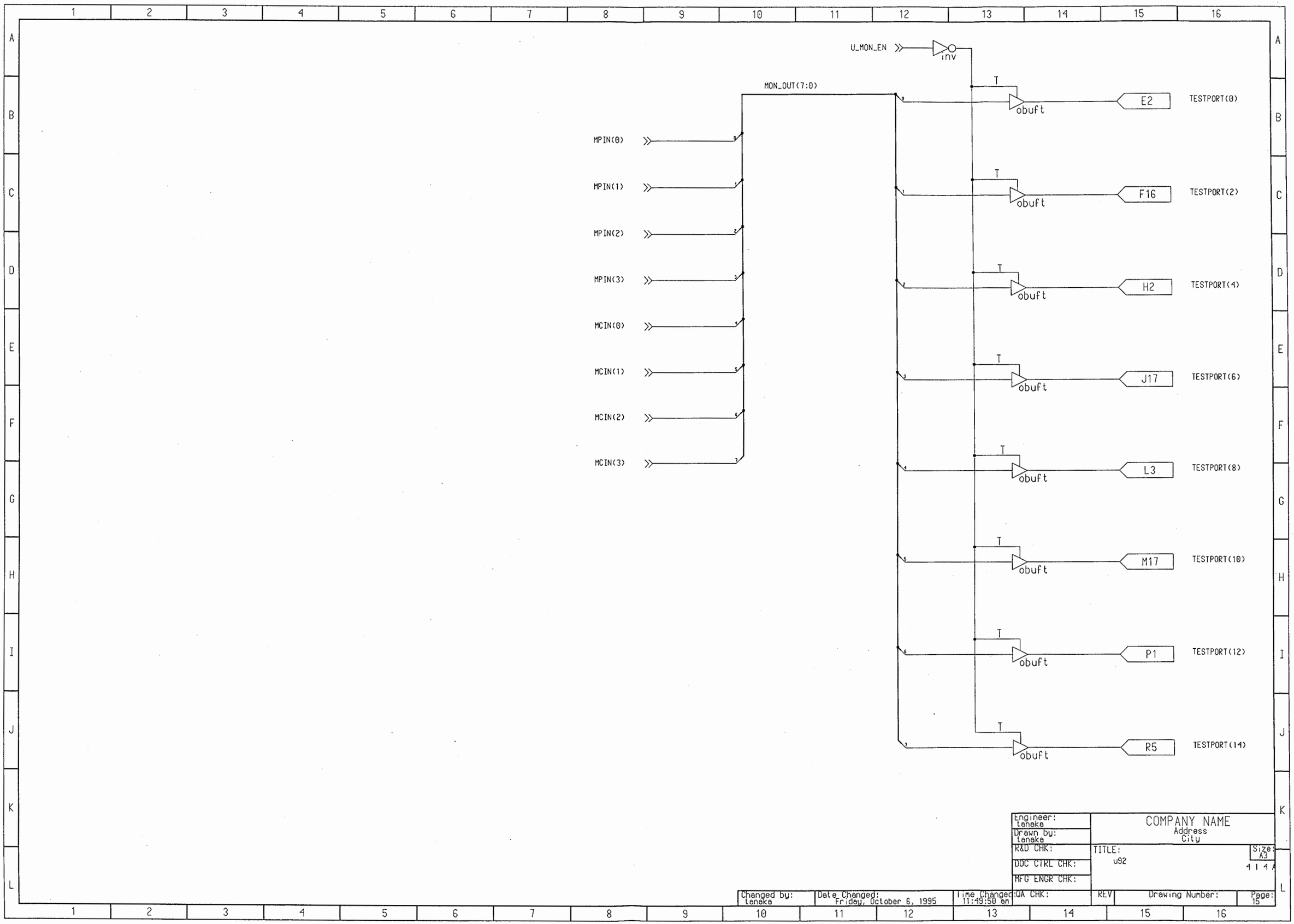
BASE +8 WN(3)I/WN(4)R REGISTER



BASE +10 WN(4)R/I REGISTER



Engineer: tanaka	COMPANY NAME
Drawn by: tanaka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: u9
MFG ENGR CHK:	Size: A2
Changed by: tanaka	Date Changed: Tuesday, September 12, 1995
Time Changed: 1:47:49 pm	QA CHK:
REV	Drawing Number:
Page: 19	



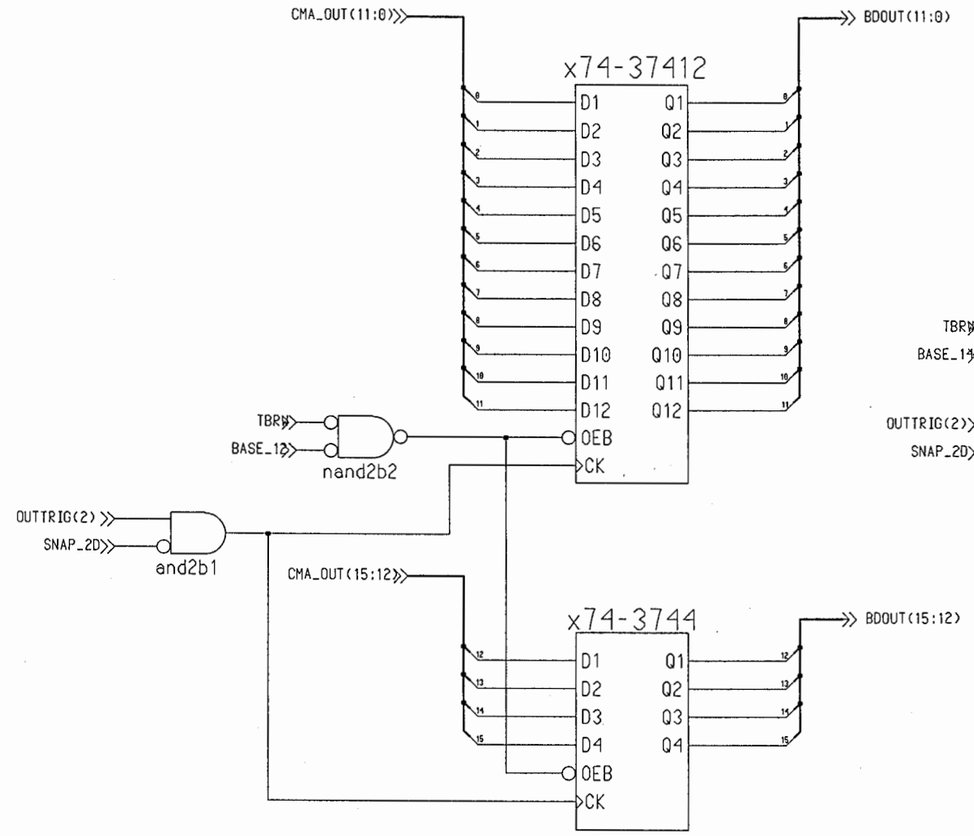
Changed by: teneke	Date Changed: Friday, October 6, 1995	Time Changed: 11:49:50 am	QA CHK:	REV	Drawing Number:

Page: 15

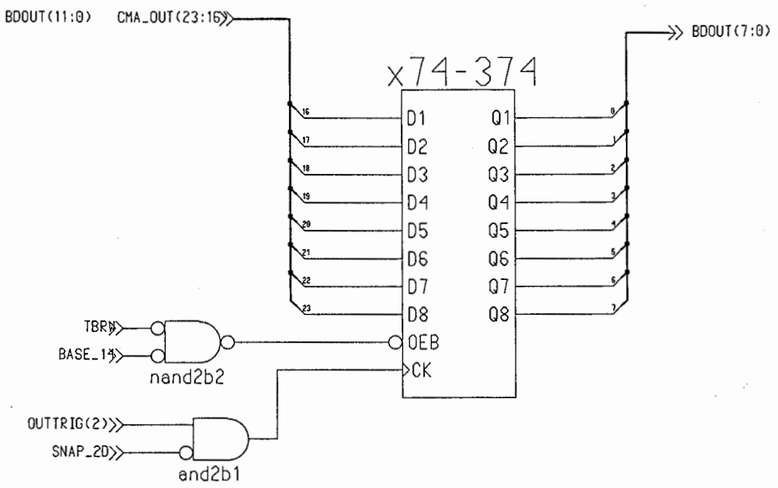
Engineer: teneke	COMPANY NAME
Drawn by: teneke	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: u92
MFG ENGR CHK:	Size: A3

4 1 4 A

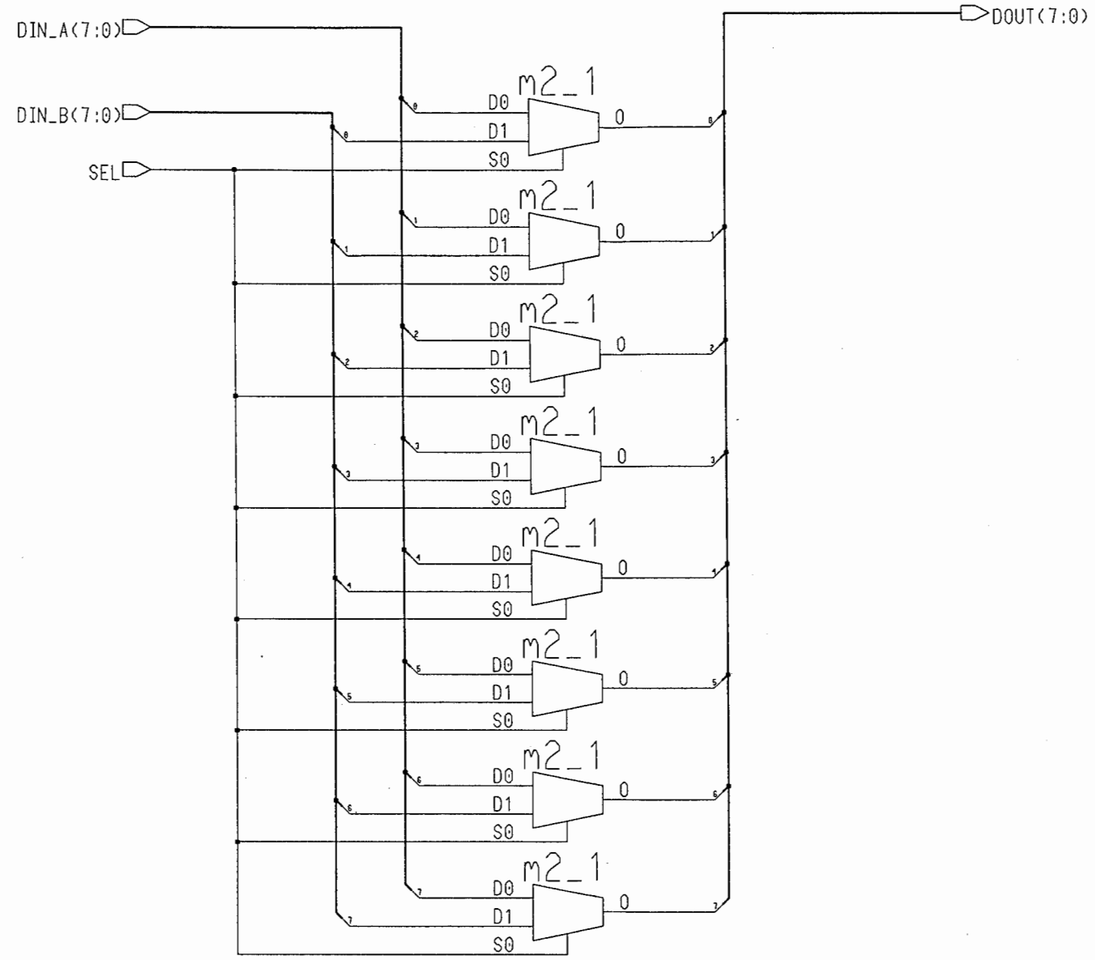
BASE +12 Yn REGISTER



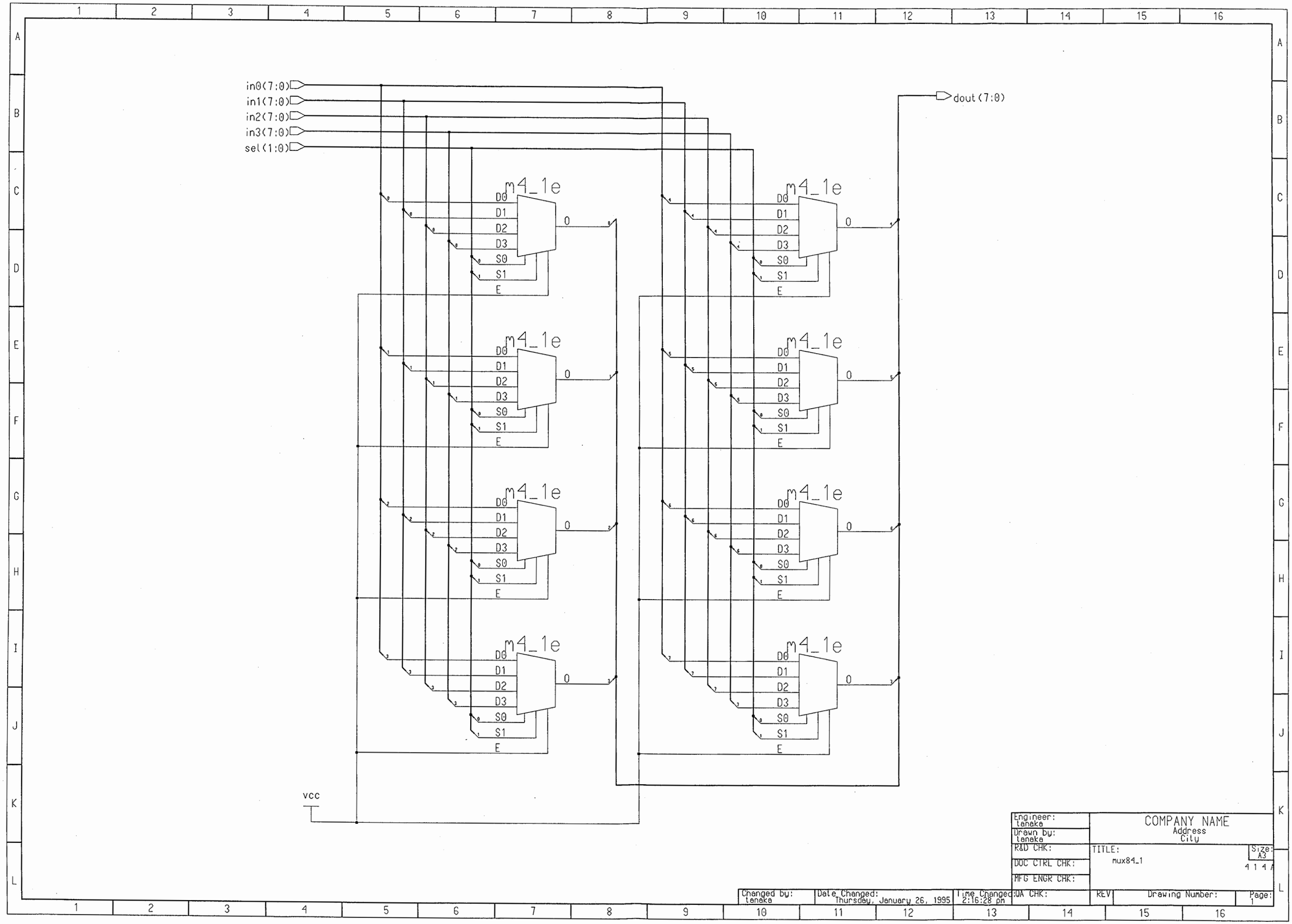
BASE +14 Yn REGISTER



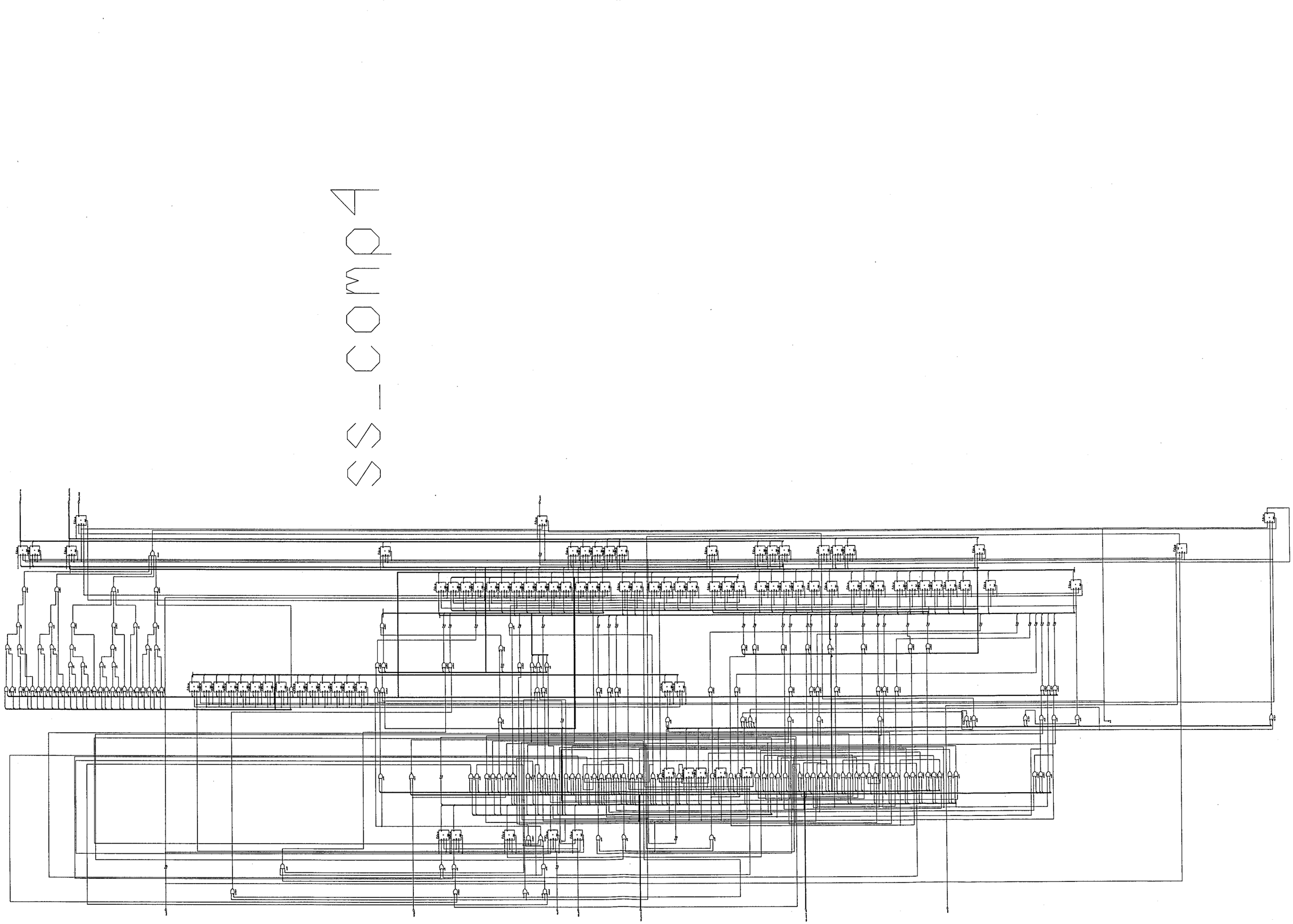
Engineer: tanaka	COMPANY NAME Address City						
Drawn by: tanaka							
R&D CHK:	TITLE: u93	Size: A3					
DOC CTRL CHK:							
MFG ENGR CHK:							
Changed by: tanaka	Date Changed: Wednesday, November 8, 1995	Time Changed: 7:33:17 pm	QA CHK:	REV	Drawing Number: 4 1 4 A	Page: 16	

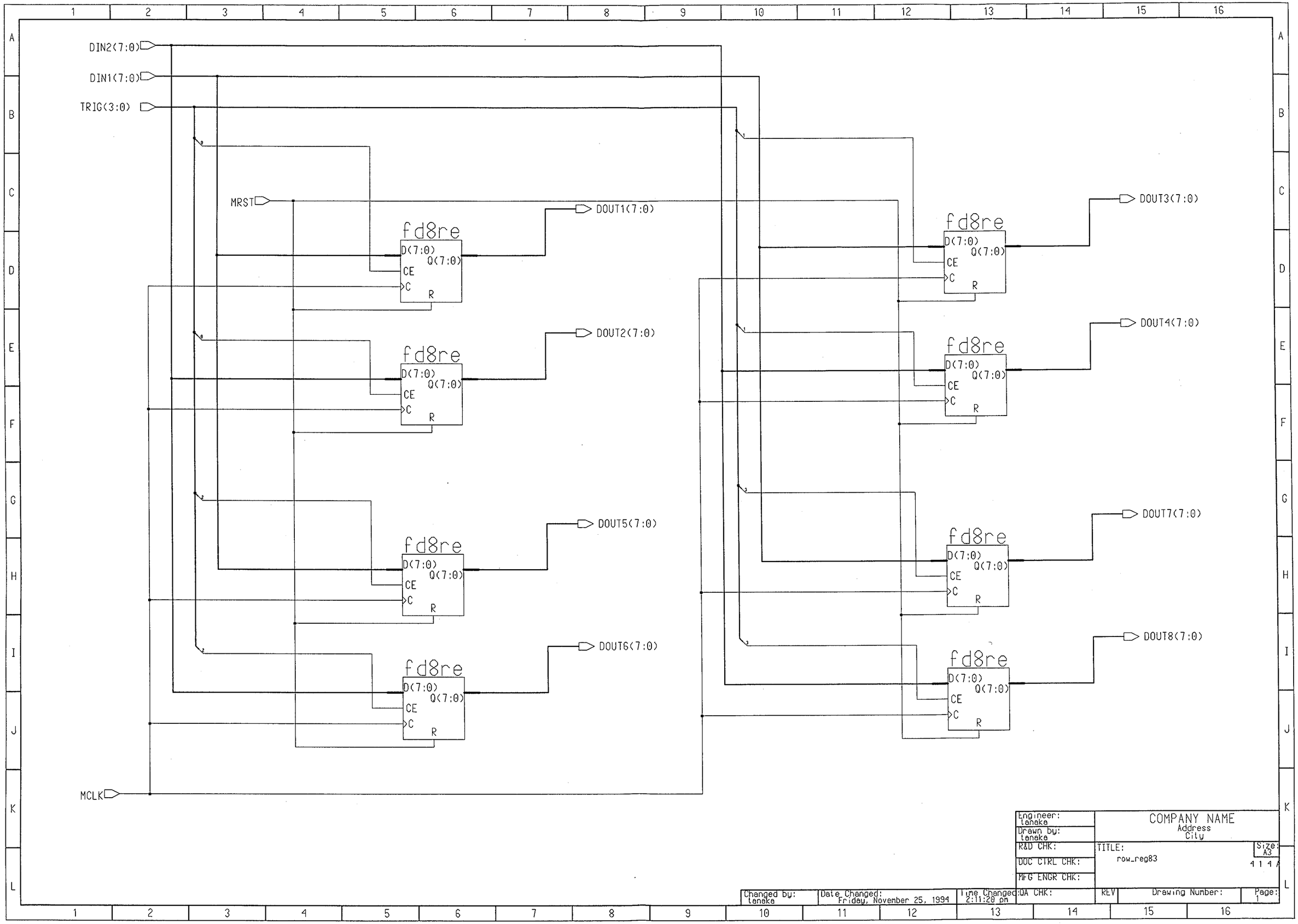


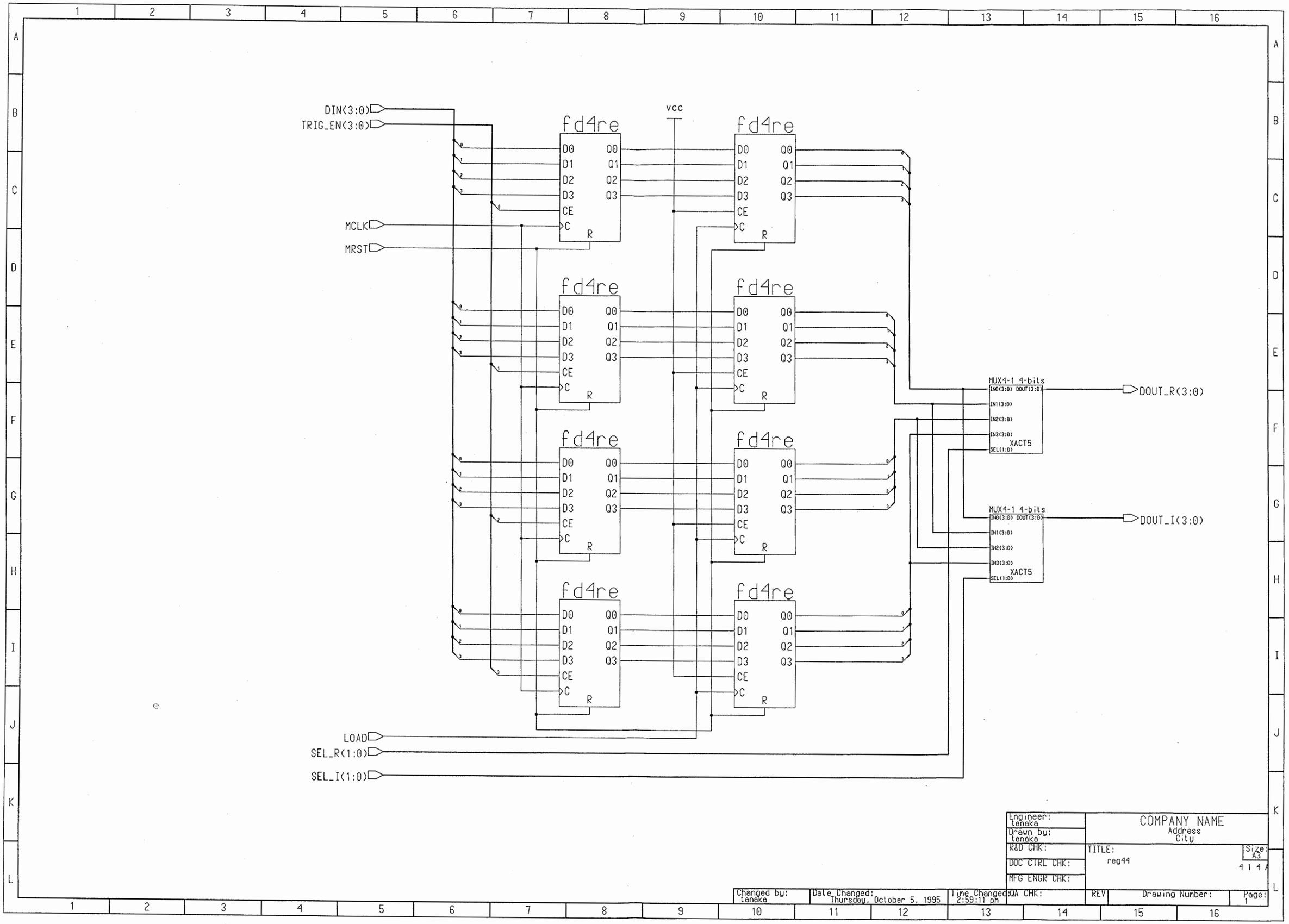
Engineer: tanaka	COMPANY NAME Address City				
Drawn by: tanaka					
R&D CHK:	TITLE: mux82_1	Size: A3			
DOC CTRL CHK:		4 1 4 A			
MFG ENGR CHK:					
Changed by: tanaka	Date Changed: Wednesday, January 25, 1995	Time Changed: 7:15:15 pm	QA CHK:	REV	Drawing Number: / Page: 1

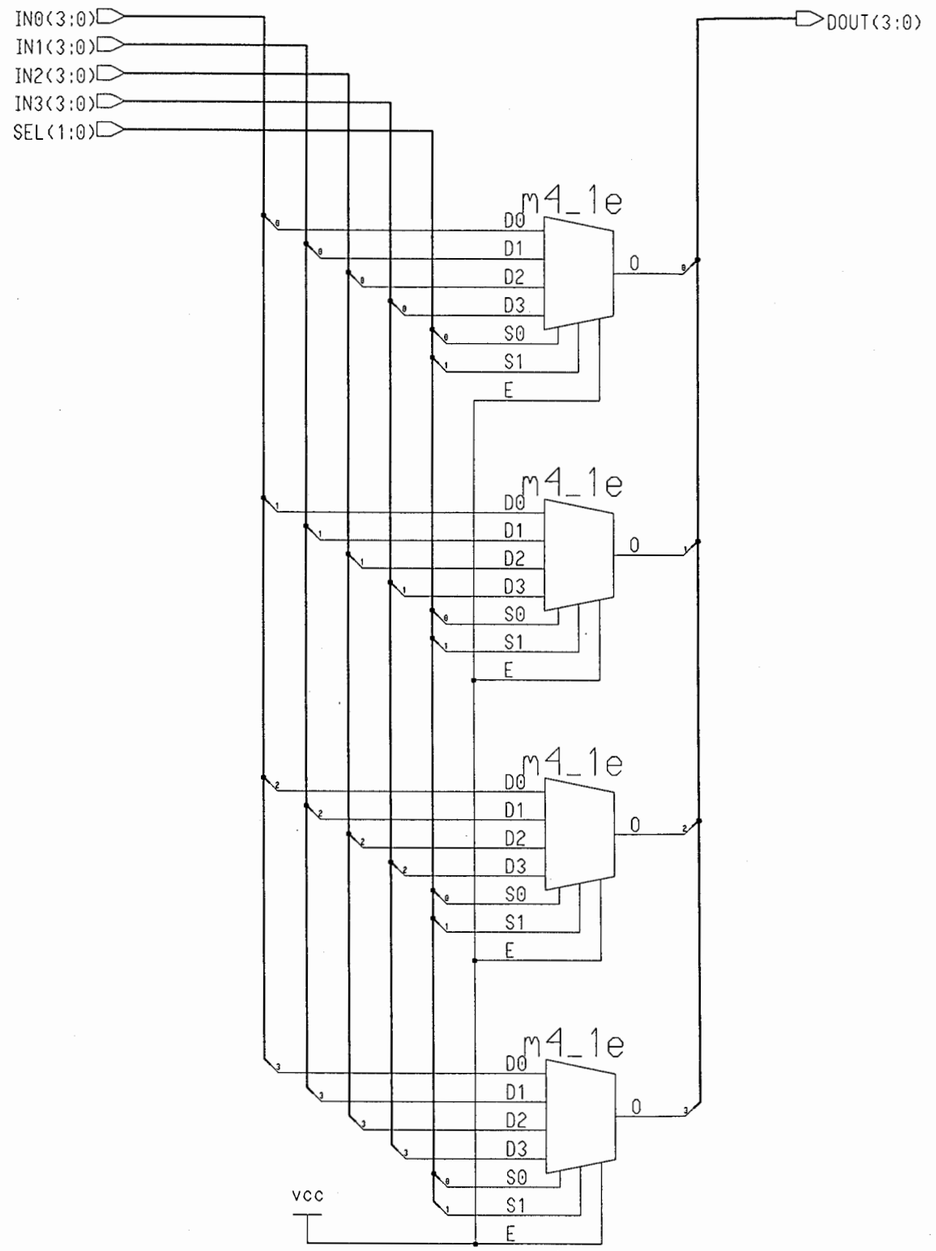

in0(7:0)
in1(7:0)
in2(7:0)
in3(7:0)
sel(1:0)
dout(7:0)
m4_1e
D0
D1
D2
D3
S0
S1
E
O
vcc
Engineer: taneka
Drawn by: taneka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
mux84_1
Size: A3
4 1 4 A
Changed by: taneka
Date Changed: Thursday, January 26, 1995
Time Changed: 2:16:28 pm
QA CHK:
REV
Drawing Number:
Page: 1

Adder_comb



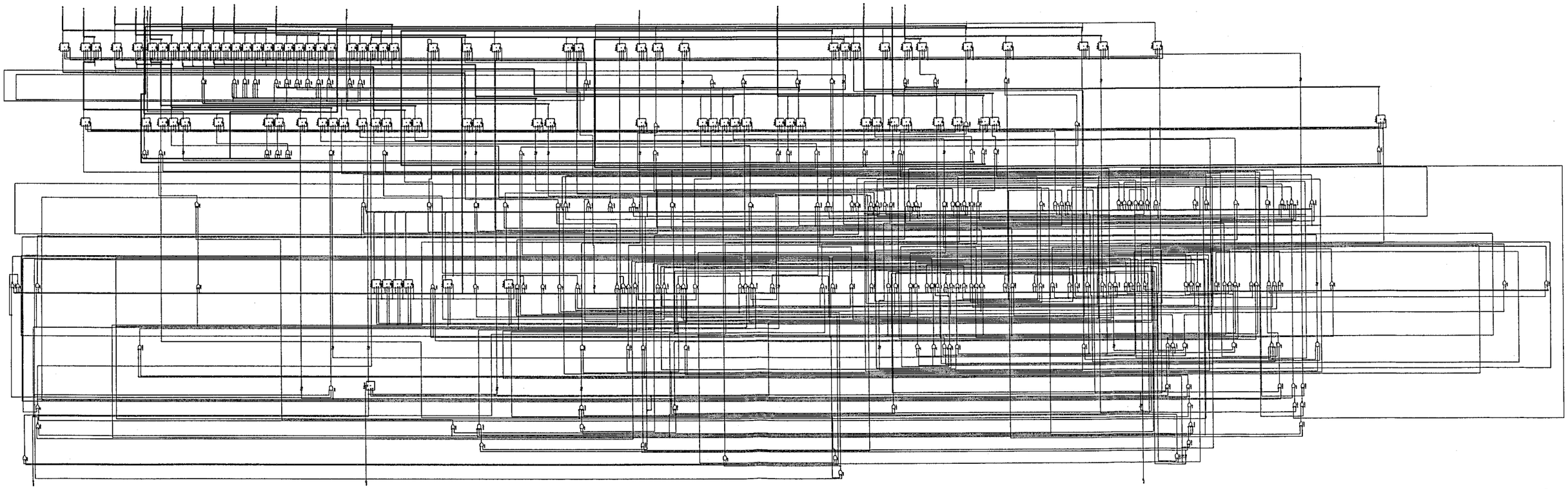

DIN2(7:0)
DIN1(7:0)
TRIG(3:0)
MRST
MCLK
fd8re
D(7:0)
Q(7:0)
CE
C
R
DOUT1(7:0)
DOUT2(7:0)
DOUT3(7:0)
DOUT4(7:0)
DOUT5(7:0)
DOUT6(7:0)
DOUT7(7:0)
DOUT8(7:0)
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
row_reg83
Size: A3
Changed by: tanaka
Date Changed: Friday, November 25, 1994
Time Changed: 2:11:20 pm
QA CHK:
REV
Drawing Number:
Page: 1

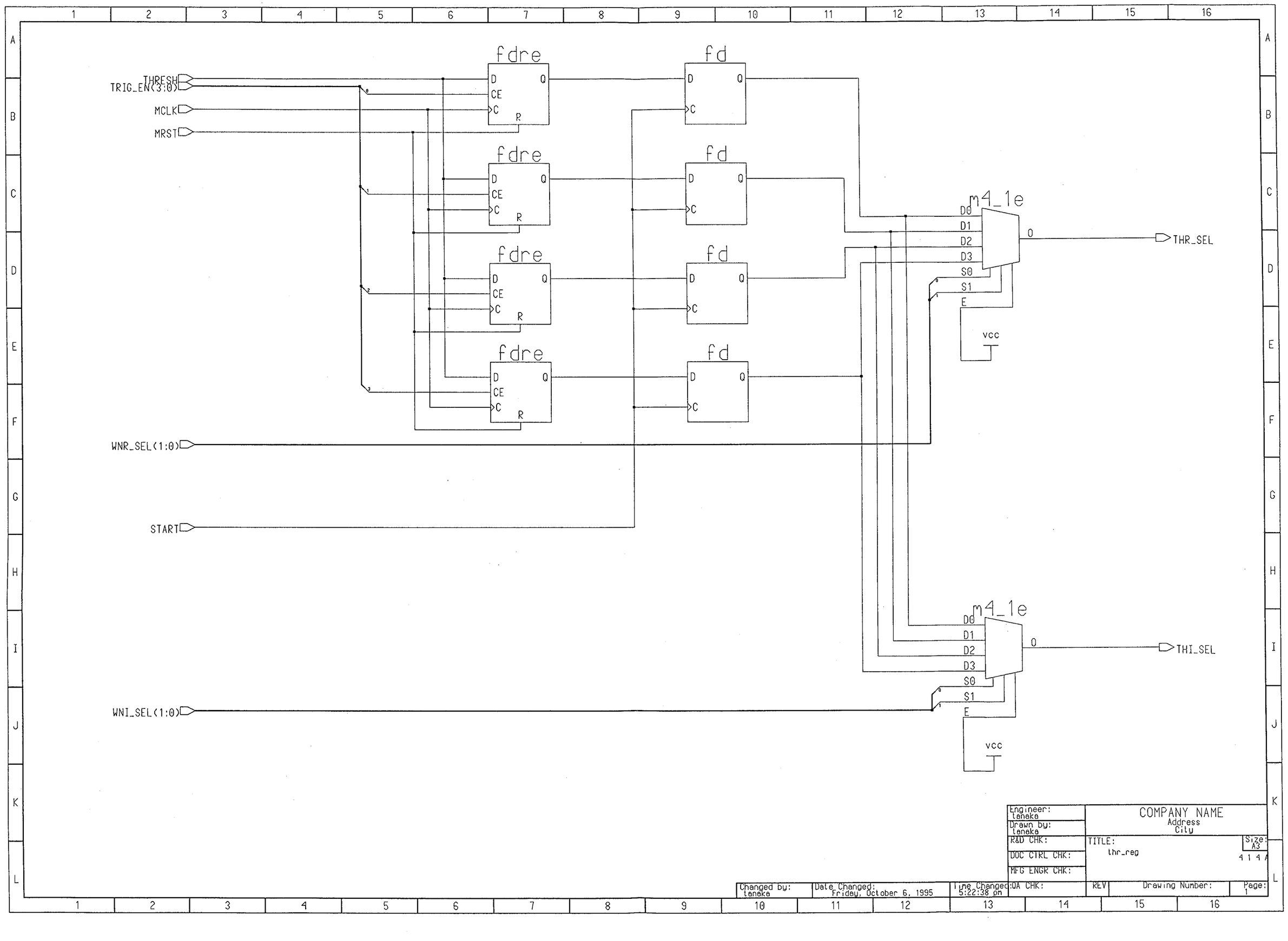

DIN(3:0)
TRIG_EN(3:0)
MCLK
MRST
LOAD
SEL_R(1:0)
SEL_I(1:0)
vcc
fd4re
D0
D1
D2
D3
CE
C
R
Q0
Q1
Q2
Q3
MUX4-1 4-bits
IN0(3:0)
IN1(3:0)
IN2(3:0)
IN3(3:0)
SEL(1:0)
DOUT(3:0)
XACT5
DOUT_R(3:0)
DOUT_I(3:0)
Engineer: tenaka
Drawn by: tenaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
QA CHK:
COMPANY NAME
Address
City
TITLE:
reg44
Size: A3
REV
Drawing Number:
Page: 1
Changed by: tenaka
Date Changed: Thursday, October 5, 1995
Time Changed: 2:59:11 pm

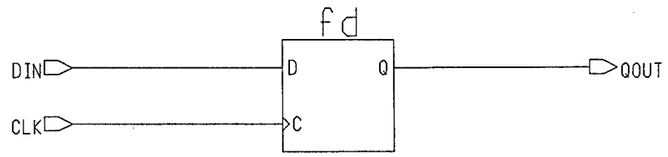


Changed by: tanaka	Date Changed: Monday, February 20, 1995	Time Changed: 7:34:56 pm	QA CHK:	REV	Drawing Number:	Page: 1
Engineer: tanaka	COMPANY NAME Address City					
Drawn by: tanaka						
R&D CHK:	TITLE: mux44_1	Size: A3				
DOC CTRL CHK:		4 1 4 A				
MFG ENGR CHK:						

ss_u9p6

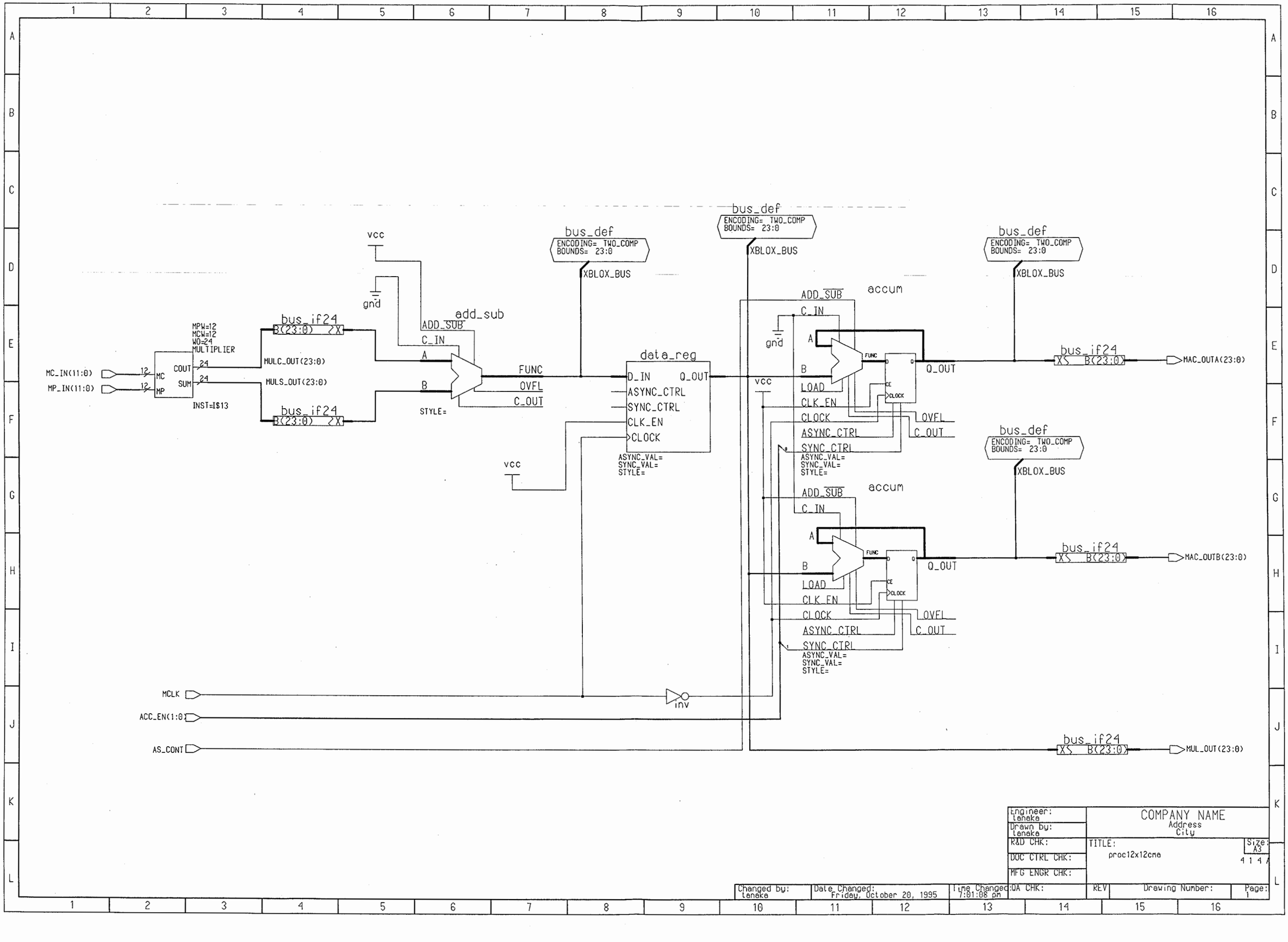



fdre
fd
m4_1e
THRESH
TRIG_EN(3:0)
MCLK
MRST
D
Q
CE
C
R
D0
D1
D2
D3
S0
S1
E
O
vcc
THR_SEL
WNR_SEL(1:0)
START
WNI_SEL(1:0)
THI_SEL
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
thr_reg
Size: A3
4 1 4 A
Changed by: tanaka
Date Changed: Friday, October 6, 1995
Time Changed: 5:22:38 pm
QA CHK:
REV
Drawing Number:
Page: 1



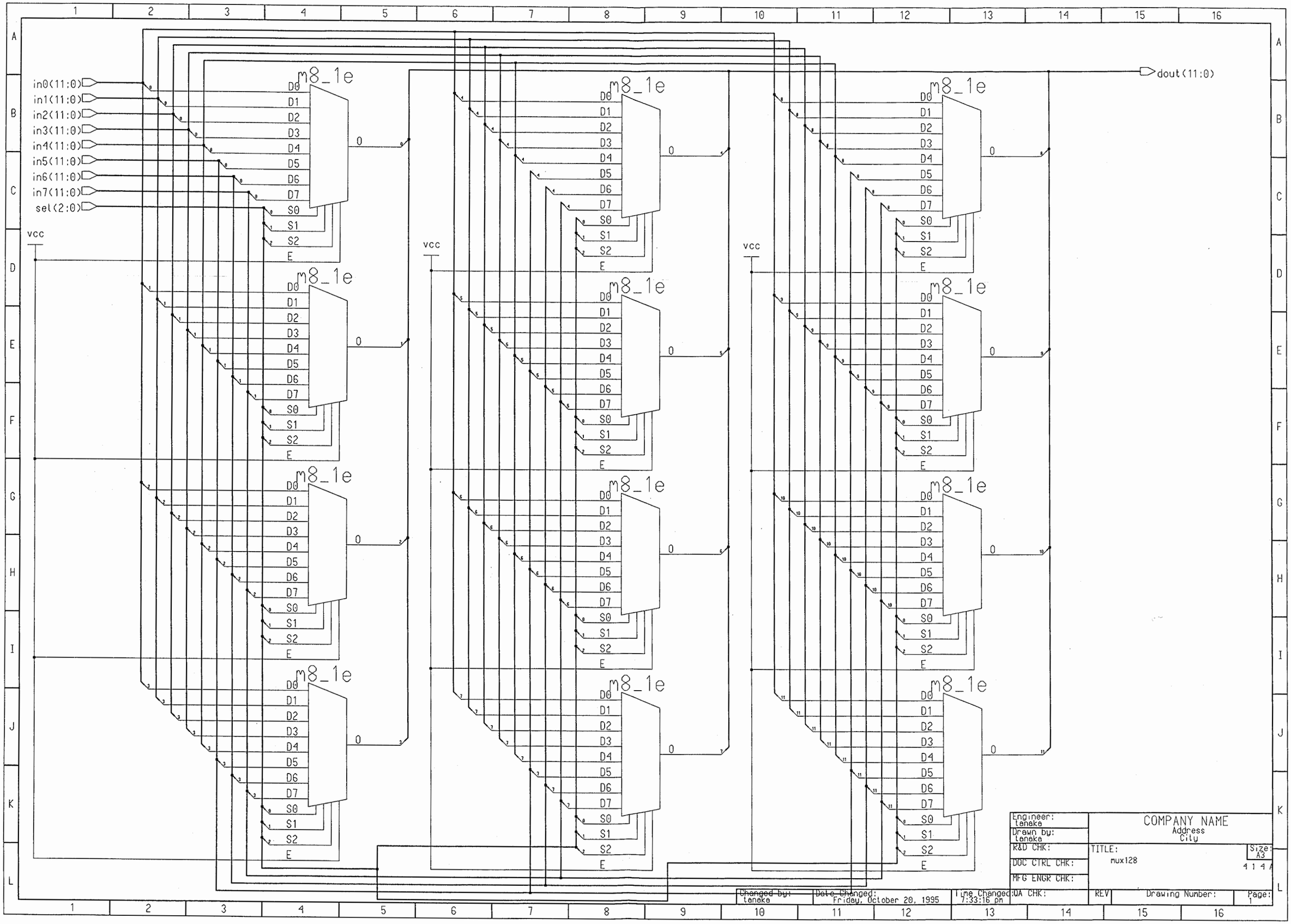
Engineer: tanaka	COMPANY NAME Address City				
Drawn by: tanaka					
R&D CHK:	TITLE: ff	Size: A3			
DOC CTRL CHK:		4 1 4 A			
MFG ENGR CHK:					
QA CHK:	REV	Drawing Number:	Page: 1		

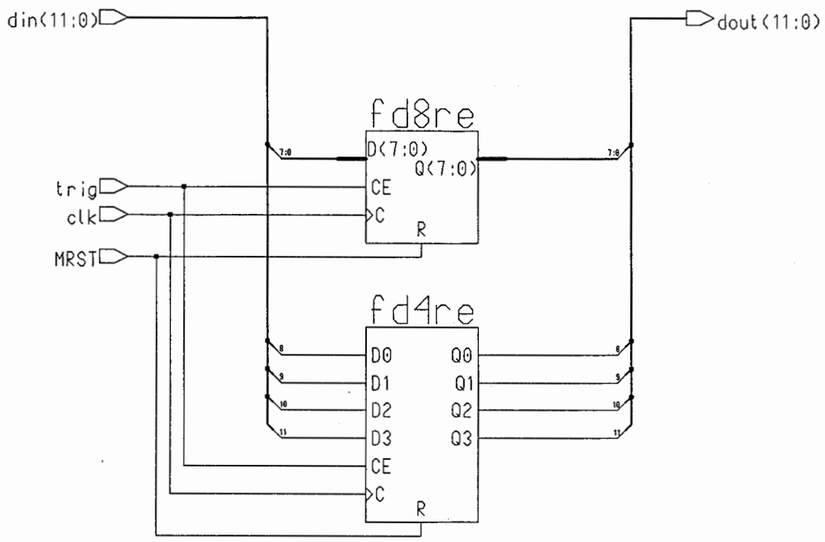
Changed by:	Date Changed:	Time Changed:
tanaka	Wednesday, June 7, 1995	6:38:14 pm



Changed by: laneke	Date Changed: Friday, October 20, 1995	Time Changed: 7:01:08 pm	QA CHK:	REV	Drawing Number:	Page: 1

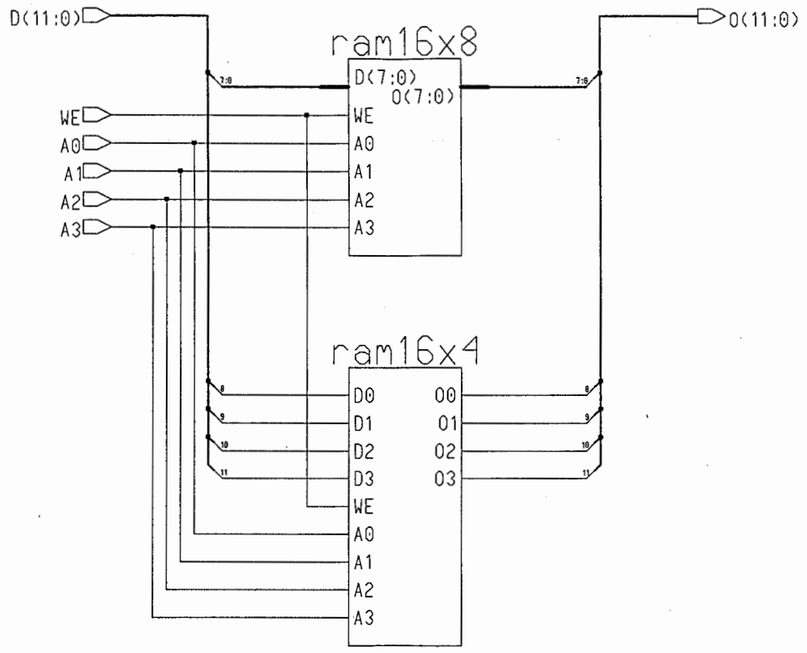
Engineer: laneke	COMPANY NAME Address City
Drawn by: laneke	
R&D CHK:	TITLE: proc12x12cma
DOC CTRL CHK:	Size: A3
MFG ENGR CHK:	4 1 4 A


in0(11:0)
in1(11:0)
in2(11:0)
in3(11:0)
in4(11:0)
in5(11:0)
in6(11:0)
in7(11:0)
sel(2:0)
vcc
m8_1e
D0
D1
D2
D3
D4
D5
D6
D7
S0
S1
S2
E
O
dout(11:0)
Engineer: taneka
Drawn by: taneka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
mux128
Size: A3
4 1 4 A
Changed by: taneka
Date Changed: Friday, October 20, 1995
Time Changed: 7:33:16 pm
QA CHK:
REV
Drawing Number:
Page: 1



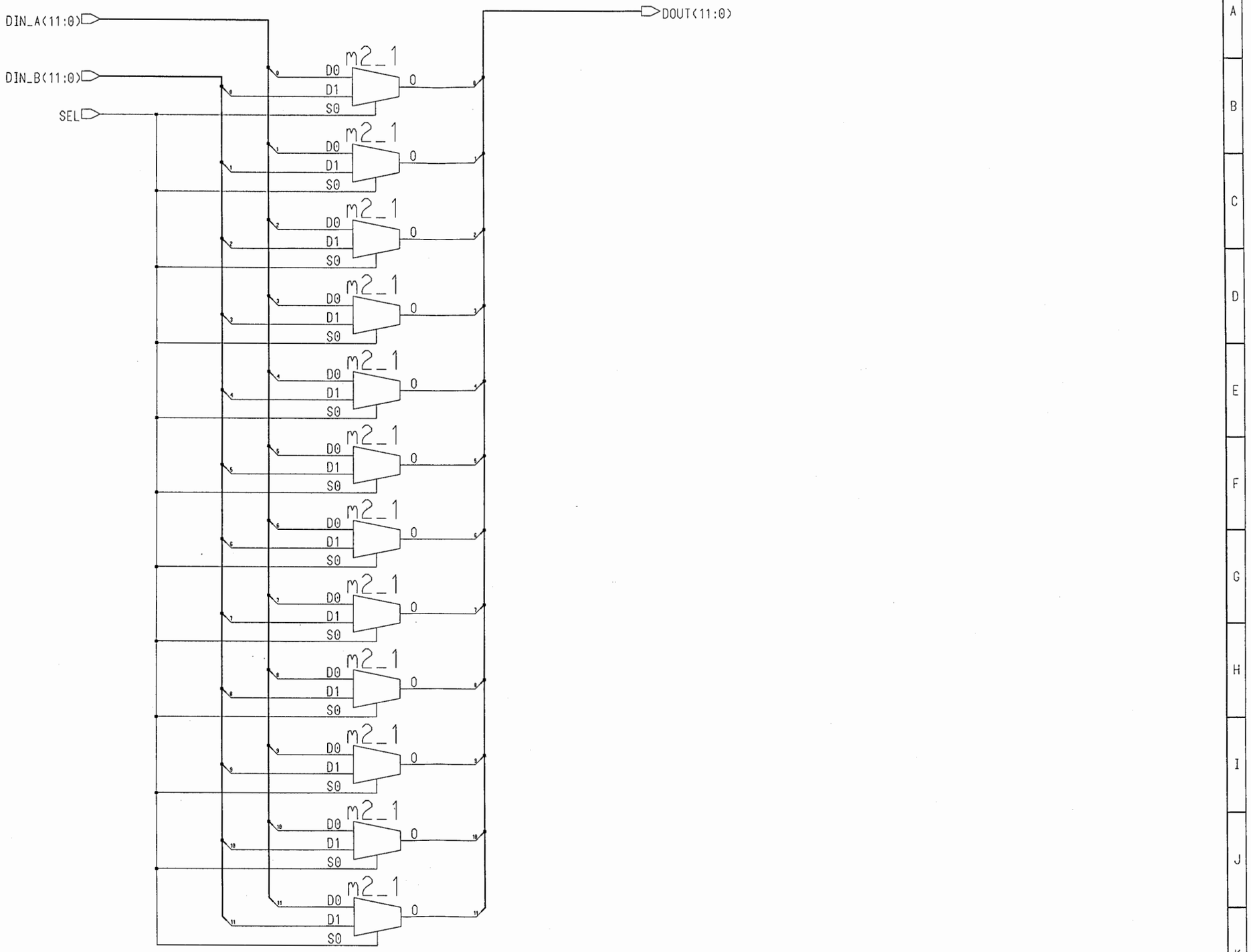
Changed by: tanaka	Date Changed: Friday, October 20, 1995	Time Changed: 8:00:45 pm			

Engineer: tanaka	COMPANY NAME Address City				
Drawn by: tanaka					
R&D CHK:	TITLE: reg12	Size: A3			
DOC CTRL CHK:					
MFG ENGR CHK:					
QA CHK:	REV	Drawing Number: 4 1 1 4 A	Page: 1		



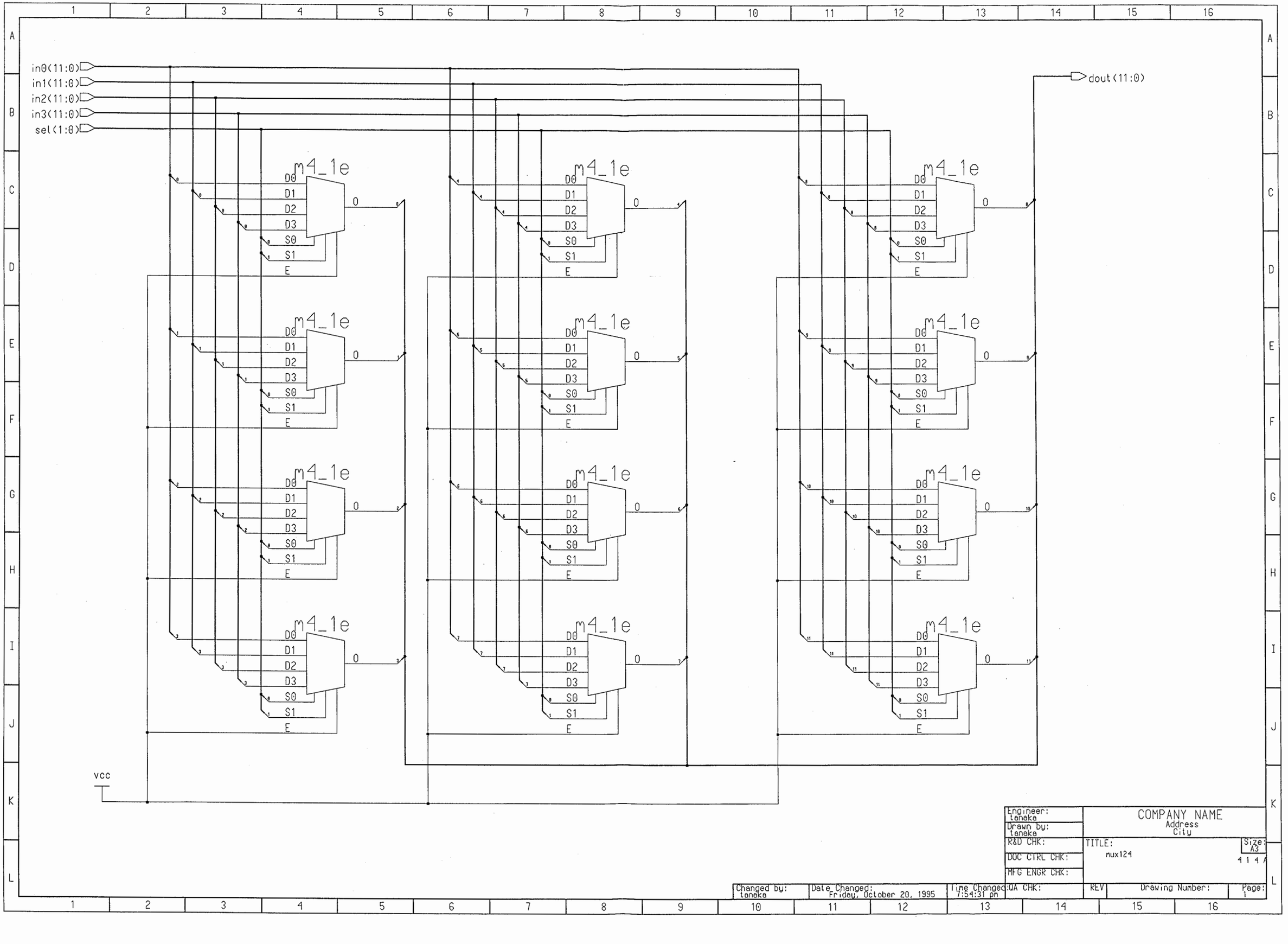
Engineer: tanaka	COMPANY NAME
Drawn by: tanaka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: ram16x12
MFG ENGR CHK:	Size: A3

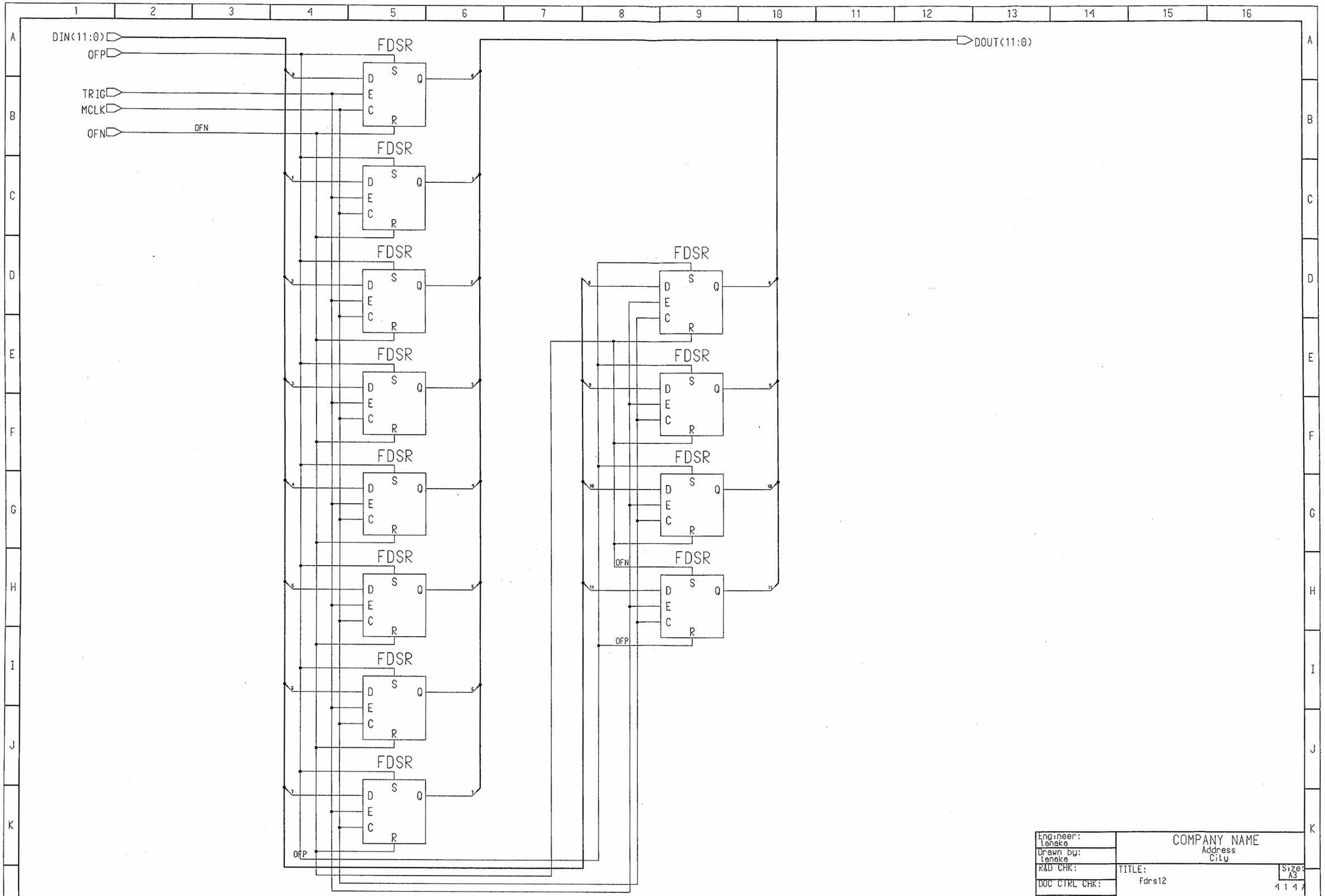
Changed by:	Date Changed:	Time Changed:	QA CHK:	REV	Drawing Number:	Page:
tanaka	Friday, October 20, 1995	8:11:49 pm				1



Changed by: tanaka	Date Changed: Friday, October 20, 1995	Time Changed: 8:06:31 pm	QA CHK:	REV	Drawing Number:

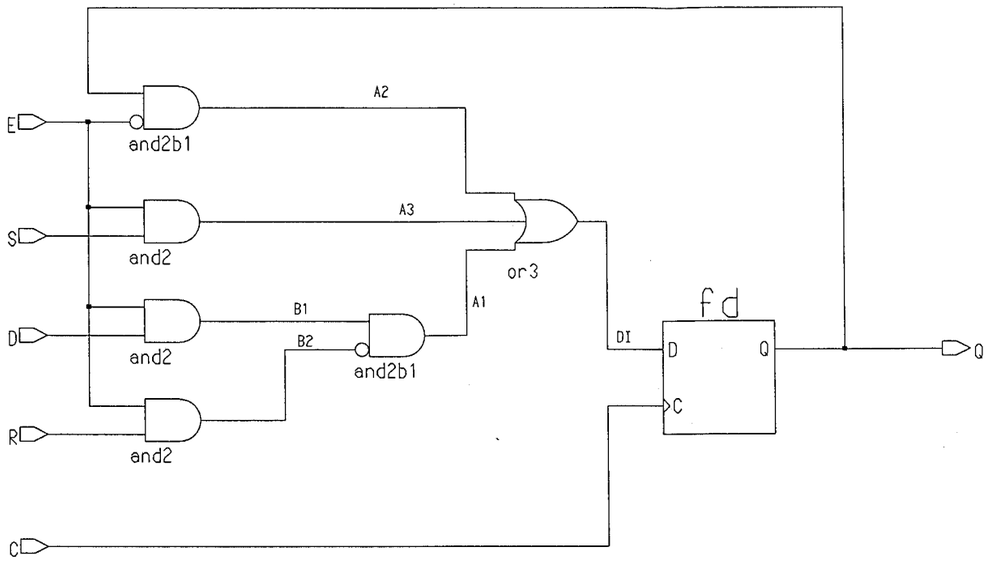
Engineer: tanaka	COMPANY NAME Address City		
Drawn by: tanaka			
R&D CHK:	TITLE: mux122	Size: A3	Page: 1
DOC CTRL CHK:		4 1 4 7	
MFG ENGR CHK:			


in0(11:0)
in1(11:0)
in2(11:0)
in3(11:0)
sel(1:0)
dout(11:0)
m4_1e
D0
D1
D2
D3
S0
S1
E
O
vcc
Engineer: taneka
Drawn by: taneka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
QA CHK:
COMPANY NAME
Address
City
TITLE:
mux124
Size: A3
4 1 4 A
Changed by: taneka
Date Changed: Friday, October 20, 1995
Time Changed: 7:54:31 pm
REV
Drawing Number:
Page: 1



Engineer: teneke	COMPANY NAME
Drawn by: teneke	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: fdrs12
MFG ENGR CHK:	Size: A3
	4 1 4 A

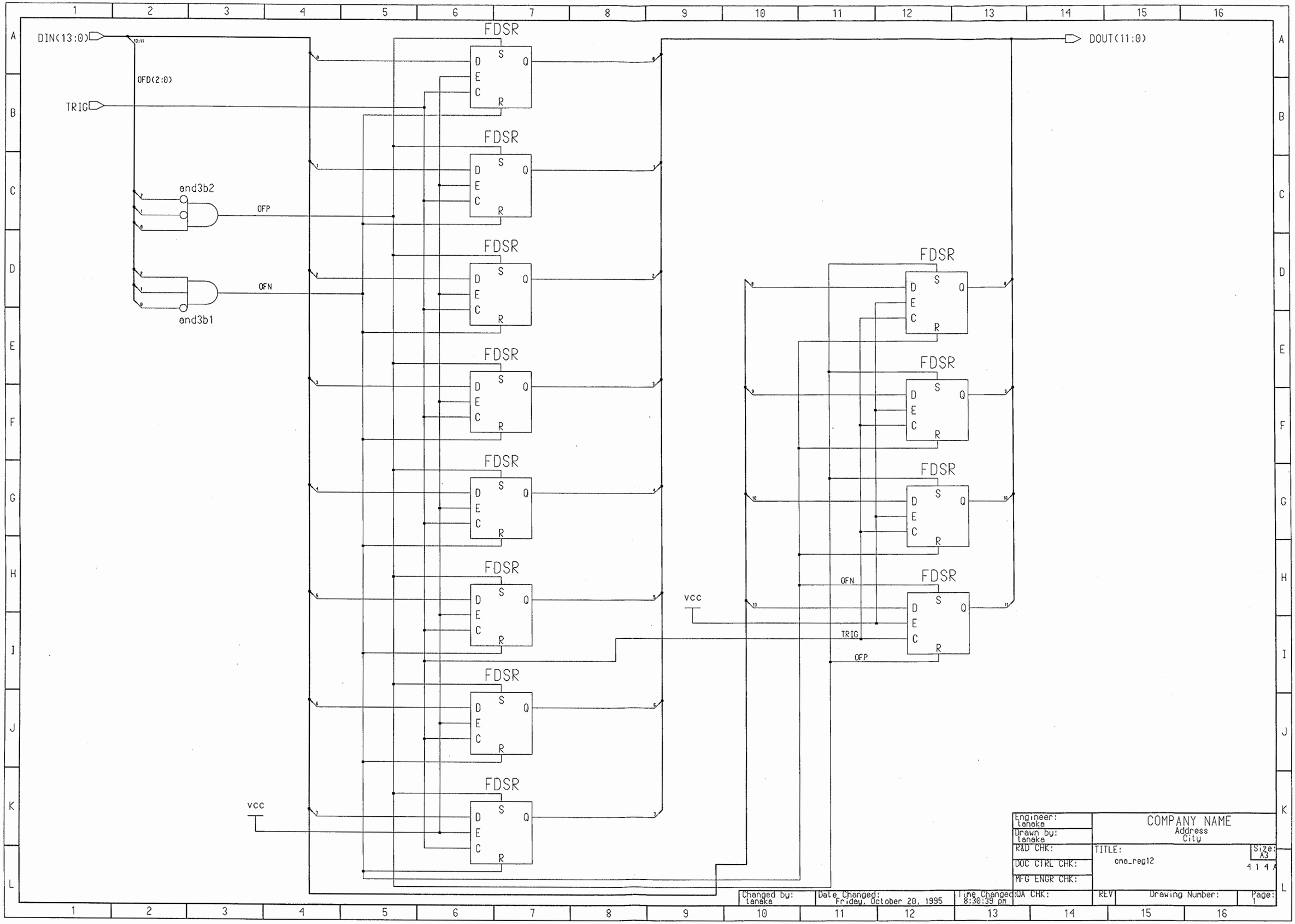
Changed by: teneke	Date Changed: Friday, October 20, 1995	Time Changed: 8:20:35 pm	QA CHK:	REV	Drawing Number:	Page: 1

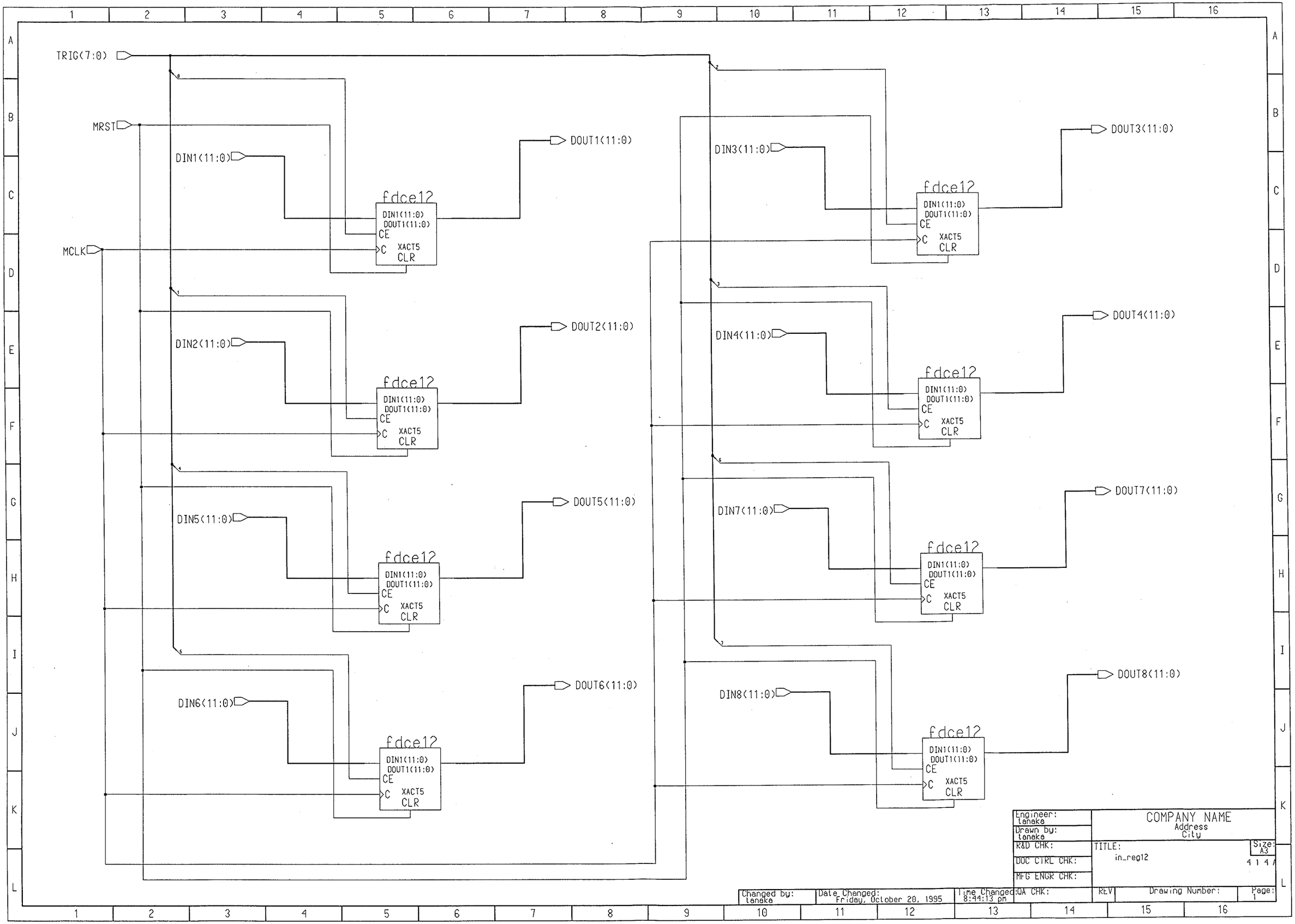


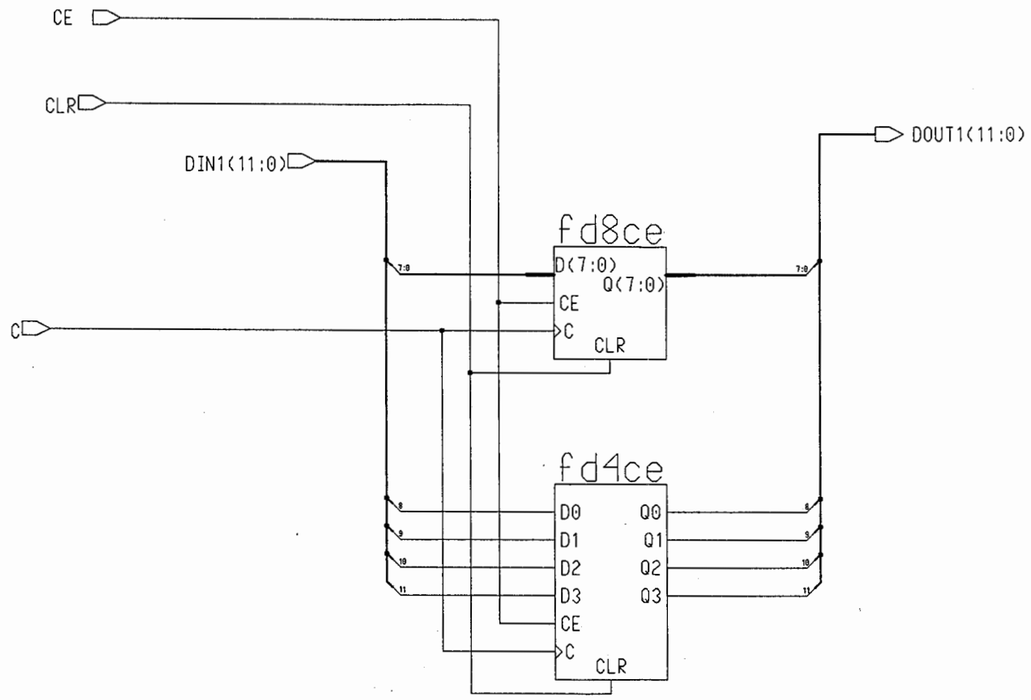
Changed by: tanaka	Date Changed: Thursday, April 27, 1995	Time Changed: 1:42:49 pm	QA CHK:	REV	Drawing Number:

Engineer: tanaka	COMPANY NAME
Drawn by: tanaka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: FDRS
MFG ENGR CHK:	Size: A3
	Page: 1

4 1 4 A


DIN(13:0)
13:11
OFD(2:0)
TRIG
and3b2
OFP
OFN
and3b1
FDSR
S
D
E
C
R
Q
DOUT(11:0)
vcc
OFN
TRIG
OFP
Engineer: taneke
Drawn by: taneke
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
QA CHK:
COMPANY NAME
Address
City
TITLE:
cne_reg12
Size: A3
4 1 4 A
REV
Drawing Number:
Page: 1
Changed by: taneke
Date Changed: Friday, October 20, 1995
Time Changed: 8:30:39 pm

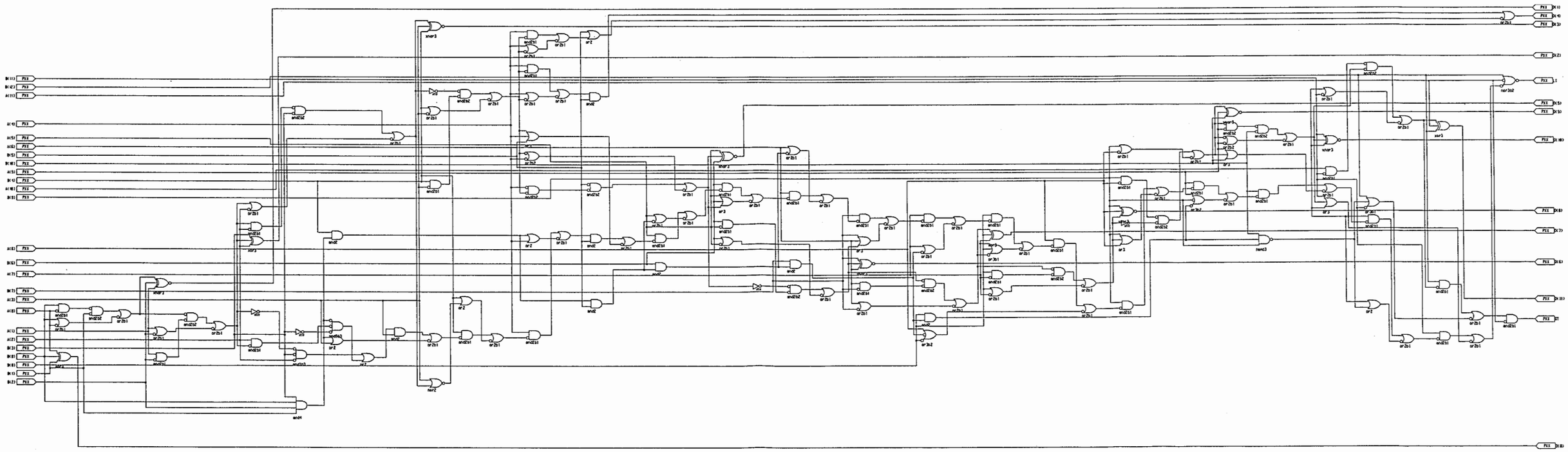

TRIG(7:0)
MRST
MCLK
DIN1(11:0)
DIN2(11:0)
DIN3(11:0)
DIN4(11:0)
DIN5(11:0)
DIN6(11:0)
DIN7(11:0)
DIN8(11:0)
DOUT1(11:0)
DOUT2(11:0)
DOUT3(11:0)
DOUT4(11:0)
DOUT5(11:0)
DOUT6(11:0)
DOUT7(11:0)
DOUT8(11:0)
fdce12
DIN1(11:0)
DOUT1(11:0)
CE
C XACT5
CLR
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
in_reg12
Size: A3
4 1 4 A
Changed by: tanaka
Date Changed: Friday, October 20, 1995
Time Changed: 8:44:13 pm
QA CHK:
REV
Drawing Number:
Page: 1

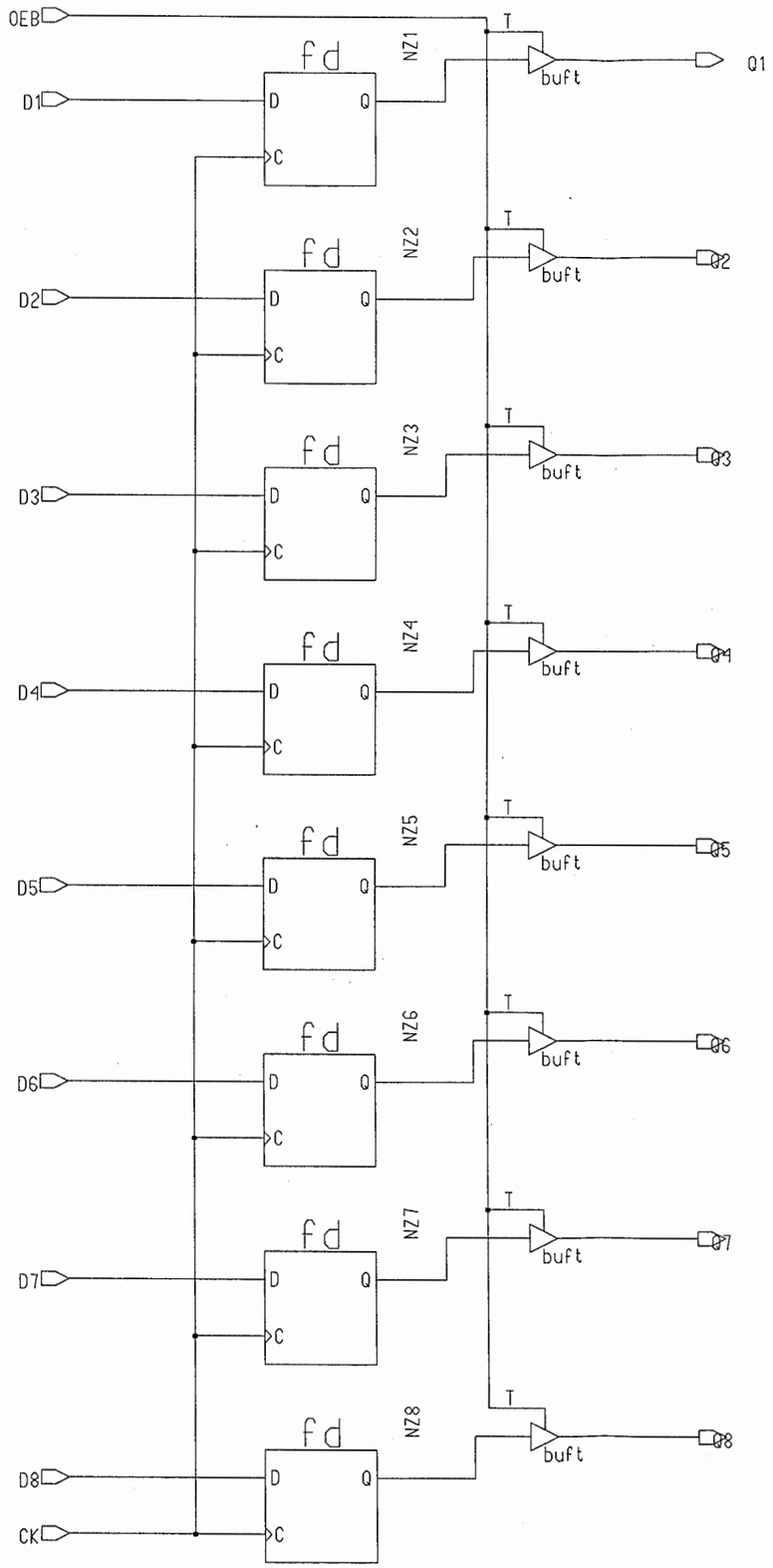


Changed by: tanaka	Date Changed: Friday, October 20, 1995	Time Changed: 8:45:42 pm	QA CHK:	REV	Drawing Number:	Page: 1

Engineer: tanaka	COMPANY NAME
Drawn by: tanaka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: fdce12
MFG ENGR CHK:	Size: A3

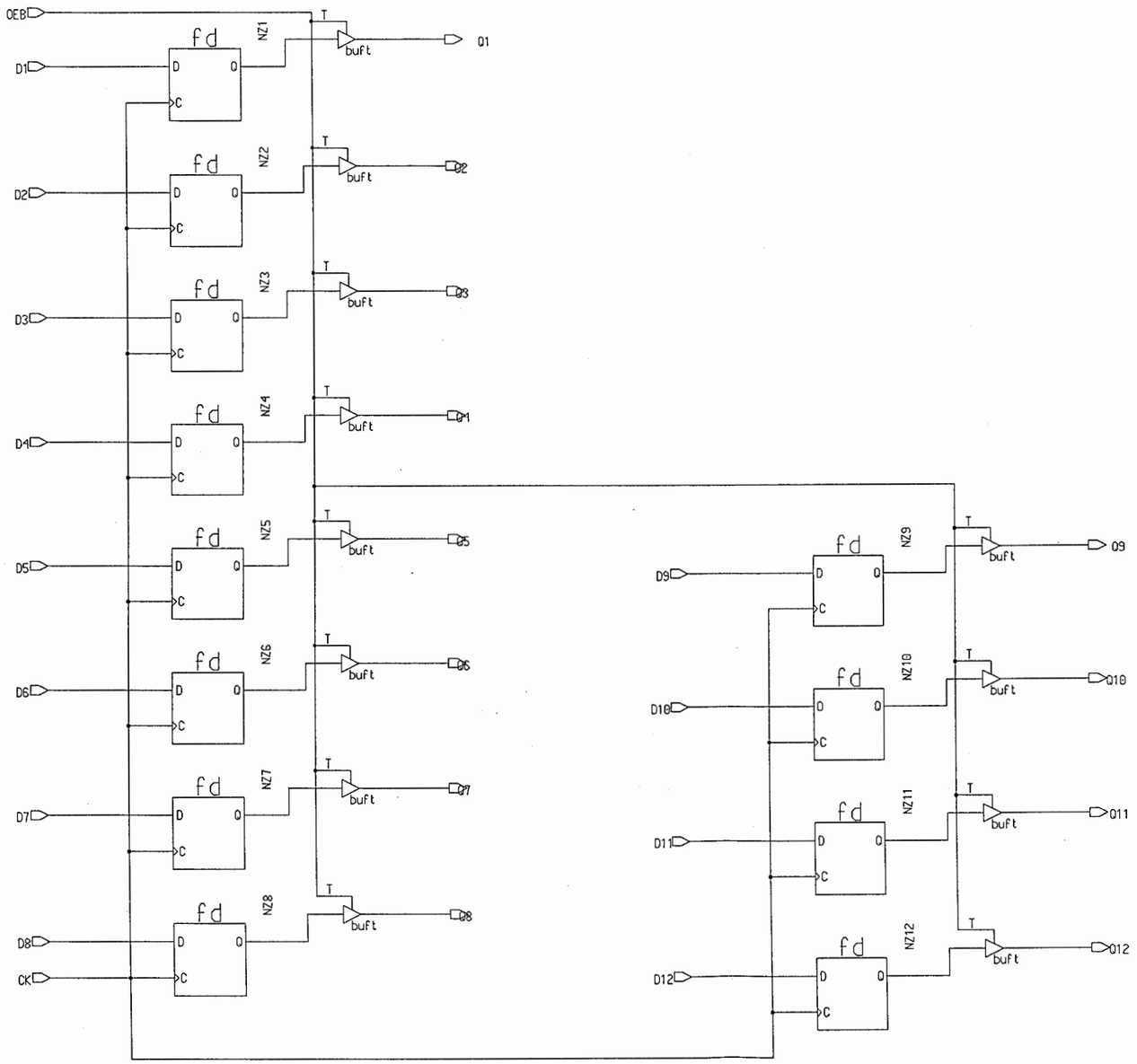
sub13of





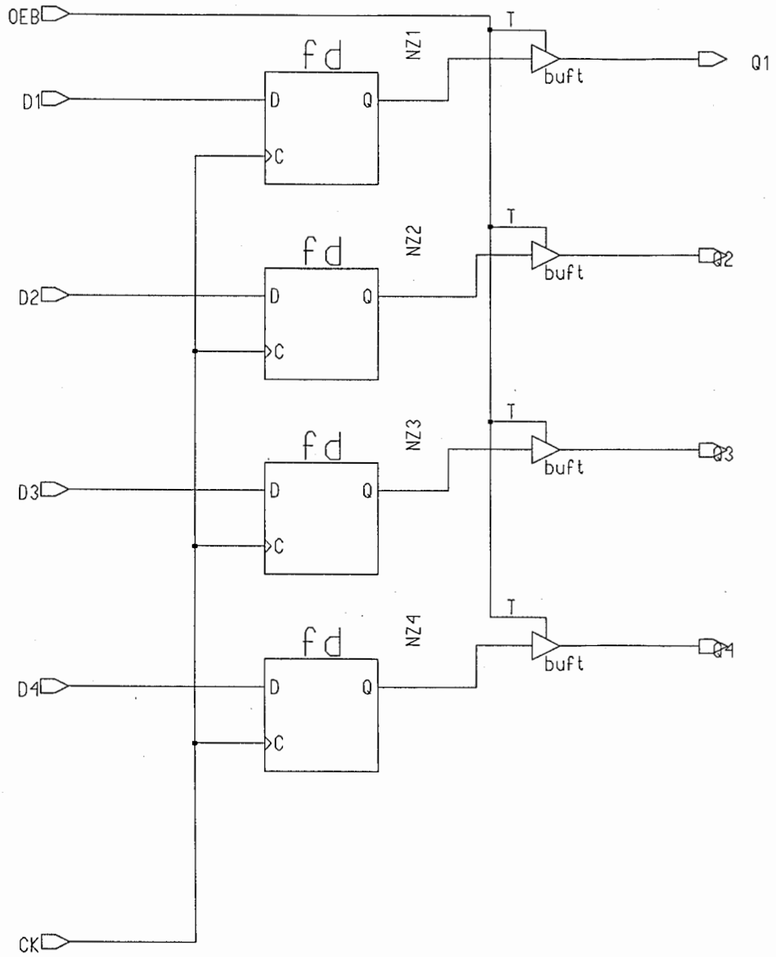
Engineer: taneke	COMPANY NAME
Drawn by: taneke	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: x74_374
MFG ENGR CHK:	Size: A3
QA CHK:	REV / Drawing Number: / Page: 1

Changed by:	Date Changed:	Time Changed:
taneke	Friday, May 19, 1995	2:29:03 pm



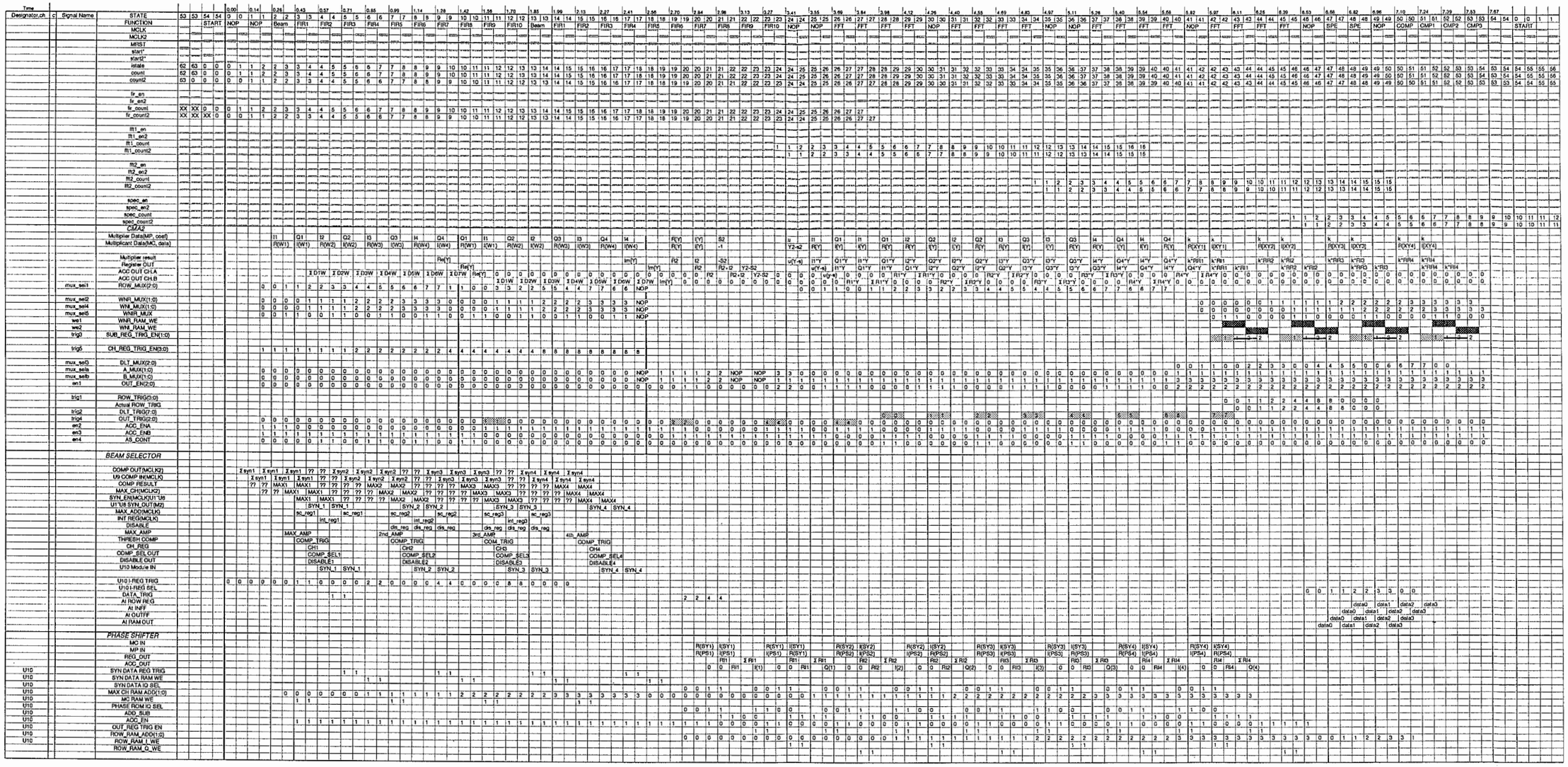
Engineer: tanaka	COMPANY NAME
Drawn by: tanaka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: x74-37412
MFG ENGR CHK:	Size: A2
QA CHK:	REV: Drawing Number: Page: 1

Changed by: tanaka | Date Changed: Wednesday, November 8, 1999 | Time Changed: 8:57:16 am



Engineer: tanaka	COMPANY NAME
Drawn by: tanaka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: x74-3744
MFG ENGR CHK:	Size: A3
QA CHK:	4 1 4 A

Changed by:	Date Changed:	Time Changed:	REV	Drawing Number:	Page:
tanaka	Wednesday, November 8, 1995	7:09:32 pm			1


Time
Designator_ch
c
Signal Name
STATE
FUNCTION
MCLK
MCLK2
MRST
start*
start2*
istate
count
count2
fir_en
fir_en2
fir_count
fir_count2
iir1_en
iir1_en2
iir1_count
iir1_count2
iir2_en
iir2_en2
iir2_count
iir2_count2
spec_en
spec_en2
spec_count
spec_count2
CMA2
Multiplier Data(MP_coef)
Multiplicant Data(MC_data)
Multiplier result
Register OUT
ACC OUT CH.A
ACC OUT CH.B
mux_sel1
mux_sel2
mux_sel4
mux_sel5
we1
we2
trig3
trig5
mux_sel3
mux_sel6
mux_sel8
en1
trig1
trig2
trig4
en2
en3
en4
BEAM SELECTOR
COMP OUT(MCLK2)
U9 COMP IN(MCLK)
COMP RESULT
MAX_CH(MCLK2)
SYN_EN(MCLK)U1~U6
U1~U6 SYN_OUT(MC)
MAX_ADD(MCLK)
INT REG(MCLK)
DISABLE
MAX_AMP
THRESH COMP
CH_REG
COMP_SEL OUT
DISABLE OUT
U10 Module IN
U10 I-REG TRIG
U10 I-REG SEL
DATA_TRIG
AI ROW REG
AI INFF
AI OUTFF
AI RAM OUT
PHASE SHIFTER
MC IN
MP IN
REG_OUT
ACC_OUT
SYN DATA REQ TRIG
SYN DATA RAM WE
SYN DATA IQ SEL
MAX CH RAM ADD(1:0)
MC RAM WE
PHASE ROM IQ SEL
ADD_SUB
ACC_EN
OUT_REG TRIG EN
ROW_RAM_ADD(1:0)
ROW_RAM_I_WE
ROW_RAM_Q_WE
U10
U10
U10
U10
U10
U10
U10
U10
U10
U10
U10
U10